A KNOWLEDGE FRAMEWORK FOR INTEGRATING MULTIPLE PERSPECTIVES IN DECISION-CENTRIC DESIGN

A Dissertation
Presented to
The Academic Faculty

by

Gregory M. Mocko

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mechanical Engineering

Georgia Institute of Technology
May 2006

A KNOWLEDGE FRAMEWORK FOR INTEGRATING MULTIPLE PERSPECTIVES IN DECISION-CENTRIC DESIGN

Approved by:

Dr. Farrokh Mistree, Co-Advisor
Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. David W. Rosen, Co-Advisor
Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Janet K. Allen
Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Christiaan J.J. Paredis
Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Nelson C. Baker
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Leon F. McGinnis
School of Industrial and Systems Engineering
Georgia Institute of Technology

Date Approved: April 10, 2006

DEDICATION

To my wife, Hannah

for her patience, perseverance, sacrifice, and support
but most of all for her unending love

ACKNOWLEDGEMENTS

The road has been long and winding and the completion of this dissertation would not have been possible without the tremendous support and encouragement from many people. First of all, I would like to thank my Ph.D. advisors Farrokh Mistree and David Rosen for their insight and continuous support over the past few years. I am very fortunate to have Farrokh and Dave as my advisors because they bring such different views and thoughts to this research. Their different personalities have given me a broader understanding and appreciation of technical research, scholarly contributions, and advising. I have learned a tremendous amount from them and feel as though they have provided me with the resources to be successful and continue to learn as a faculty member.

Farrokh has been an inspiration to me. His enthusiasm for education and interest in students is amazing. I have learned so much from him, He has helped me to understand what scholarly research really is and how to view the "big picture". Farrokh has been a tremendous mentor to me. Dave Rosen has helped me to identify my research contributions and has provided me with a spark for continued research in the area of information modeling. I am truly amazed at his ability to listen and provide valuable feedback and direction. I have learned to be prepared, well-organized, and confident. He has added significant value to this research.

I would also like to thank my dissertation committee members Janet K. Allen, Nelson Baker, Leon McGinnis, and Chris Paredis for their feedback and comments for

improving this work and sharpening me as a researcher. Their comments have challenged and helped me to more clearly see the value in my research. I thank all the members of the Systems Realization Laboratory and Engineering Information Systems Laboratory including Chris Williams, Matt Chamberlain, Andrew Schnell, Hae-Jin Choi, Jitesh Panchal, Marco Fernandez, Manas Bajaj, Injoong Kim and others for contributing to a research environment that has proven immensely rewarding. I want to thank my friends Kristy and Trae Westmoreland who have helped to make Atlanta home and became like family and Serena Levy and Scott Harrison for encouraging me on this journey. I would like to thank my family for the support they have provided me. I thank my daughter, Stella, for her smile, laughter, and unconditional love. She is the *greatest distraction,* and without knowing she has helped to keep me sane and grounded. I want to thank my wife, Hannah, for her fortitude to endure the ups and downs and her willingness to travel all over the country in pursuit of my dreams. She cannot truly know how much I appreciate all she has done. I thank her for her love, encouragement, and support over the last many years.

I am grateful for the financial support that I have received throughout my education at Georgia Tech. I have had the opportunity to be supported and gain experience from a range of research projects in engineering. The work presented in this dissertation has been motivated and supported by Northstar Vinyl Products, Parametric Technologies Corporation (PTC), IBM, and the Rapid Prototyping and Manufacturing Institute (RPMI) member companies. Additionally, financial support has been provided by the George W. Woodruff Graduate Teaching Fellowship, the National Science Foundation (NSF)-IGERT Fellowship and TI:GER program at Georgia Tech, Georgia Tech Institution

Presidential Fellowship, and National Institute of Standards and Technology (NIST) Grant B-01-691. As a new faculty member at Clemson University, I gratefully acknowledge the Frederick Storey Endowed Fellowship from the College of Engineering at Georgia Tech for providing continued financial support.

Finally, I thank my first advisor at Georgia Tech, the late Dr. Robert E. Fulton, for taking me on as a student at Georgia Tech and exposing me to the exciting research domain of engineering information management. He was the spark that started my interest in this area and helped to me to formulate my research direction. His practical experience and application greatly supplemented my theoretical research interest.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

SUMMARY

Problem: Engineering designers often "view" product-related design information from various perspectives throughout the product realization process depending on their domain and design concerns. In addition, designers make decisions based on information from multiple sources that span various perspectives in the product realization process. However, the information is often independent, limited to a single perspective, and not formally represented making it difficult to exchange and share in the context of engineering design decisions. Despite advances in computing technology and the maturation of design support tools and product models, significant gaps exist that limit the ability to collaborate across design perspectives. Current research efforts do not adequately address information representation and exchange for engineering design decisions. Hence, the primary challenge is to develop computational representations to facilitate the exchange product-related information for engineering design decision support.

Approach: To address this challenge, our primary hypothesis is a formal language can be developed for capturing the semantics of engineering design decisions and provide a *standardized digital interface* through which design information can be exchanged across design perspectives. This primary hypothesis is realized in this dissertation as a description logic-based formal language and information model. The formal language comprises three components: (1) a computational information representation for engineering design decisions, (2) a computational information representation for

engineering analysis models, and (3) reasoning and querying services for capturing, organizing, and reusing design decision knowledge. The primary research is decomposed into four, closely related sub-questions. The first two research questions are related to the need to explicitly capture the information associated with design decisions and analysis support models. The first hypothesis used to answer this question is that formal information models can be developed for explicitly capturing the entities and relationships associated with the compromise decision support problem and associated analysis models. The second research question is focused on the realization of the graphical information model into a computer-processible representation for capturing decision semantics. The hypothesis used to answer this question is that description logic can be used as the information modeling formalism for developing computational representations. The third research question is related to the organization, retrieval, and querying of decision and analysis information. The hypothesis used to answer this question is the DL reasoning algorithms can be leveraged, such that decision information can be checked for consistency and organized in a hierarchical structure.

Validation: The information modeling approach and formal language developed in this dissertation are validated using the validation-square approach. The validation square approach consists of validating the research hypotheses from the theoretical and empirical validation. Empirical validation of the formal language is completed through several examples. The examples include explicitly modeling the information associated with the general cDSP construct, single-goal cDSP, multi-goal cDSP, Type I and Type II Robust cDSP, generic analysis model representation, computational-based and equation-based analysis models. Additionally, the formal language is validated through several

example design problems including the structural design of a cantilever beam, and multi-disciplinary design of a structural fin array heat sink for electronic cooling. Validation and verification of the formal language is achieved by systematically increasing the complexity of and the aspects considered in the example problems, thus building confidence in the formal language for general applicability in design.

Contributions: The contributions from this dissertation address issues associated with engineering information management. The specific contribution from completing the research in this dissertation is a formal language for capturing the semantics of engineering design decisions and analysis support models. The formal language consists of four components including: (1) a systematic method for formulating engineering design decisions, (2) a vocabulary of the concepts and properties associated with multi-objective design decisions, (3) a graphical representation and notation of the information models for multi-objective design decisions and analysis models, and (4) DL concept definitions based on the vocabulary that provide a computer interpretable representation. The components, collectively, provide a means for unambiguously representing and exchanging decision-related information between multiple engineering design disciplines.

CHAPTER 1:

RESEARCH FOUNDATIONS

Aims

- To establish the motivating problem scenario for integration multiple design perspective.
- To identify the design challenges associated with sharing and exchanging information in the product realization process.
- To establish the frame of reference and research foundation by examining current state of the art and relevant literature.
- To scope the research problem and clearly define the focus, approach and contributions of the research by establishing:
    - Fundamental research questions and hypothesis addressed in this dissertation.
    - Research contributions and value as a result.
- To present the validation strategy used in this research to verify and validate the research questions and contribution in light of the research hypothesis and provide a framework for proceeding with the research.

The principal objective is this research is to:

Develop a computational representation (i.e., a formal language) for capturing the semantics of engineering design decisions to facilitate the integration of information from multiple design perspectives.

Specifically, the objective in this research is to develop a computational representation of the compromise decision support problem (cDSP) and the associated analysis support models. The computational representation provides a formal language

that enables information from multiple design perspectives to be exchanged in the context of engineering design decisions.

The motivation for this research is twofold. First, decision-centric design (DCD) and multi-objective design decisions are a central component in the product realization process. Decision-centric design is an approach for representing design as set of design decisions and activities that support the decision-making process. Decision-centric design (DCD) is an overarching philosophy and mathematical approach for representing the design process in which the decisions encountered by engineering designers are formally modeled and serve as integrators between design disciplines and domains. Decisions provide a common "language" for modeling multi-disciplinary design problems [94]. The DCD community has made significant contributions to address many of the issues associated with mathematical modeling and strategies for executing complex design decisions [17; 55; 141]. However, current DCD research and development efforts have failed to adequately address information captured in the formulation of engineering design decisions (e.g., [34; 47; 67; 72; 129]). The overarching need for information exchange and data management, recognized in the early 1990's, associated with multi-disciplinary decision making remains an open research issue [3].

Second, the development of engineering information management (EIM) systems and technologies is motivated by the need to share product information across the extended enterprise. For example, standardized product information models have enabled a broad scope of product-related design information to be exchanged amongst heterogeneous design tools, including computer-aided design (CAD) and computer-aided engineering (CAE) [57; 65]. Standardized product models were originally developed for

exchanging product geometry. Recently, the scope of product models has been expanded to capture additional product information including: product family information, design evolution and rationale, design and analysis integration, behavioral models, and function-based design representations [23; 24; 43-45; 59; 74; 75; 97; 112]. However, information models for capturing the semantics of engineering design decisions have not received adequate attention.

In summary, DCD provides a philosophical approach for representing the product realization process as a set of design decision using mathematical constructs [164]. Similarly, the EIM community has addressed the need for developing computational representations of engineering design information. However, synergistic development efforts focused on information models of design decisions have not come to fruition. The resulting gaps associated with information management for engineering design decisions are summarized in Figure 1-1.

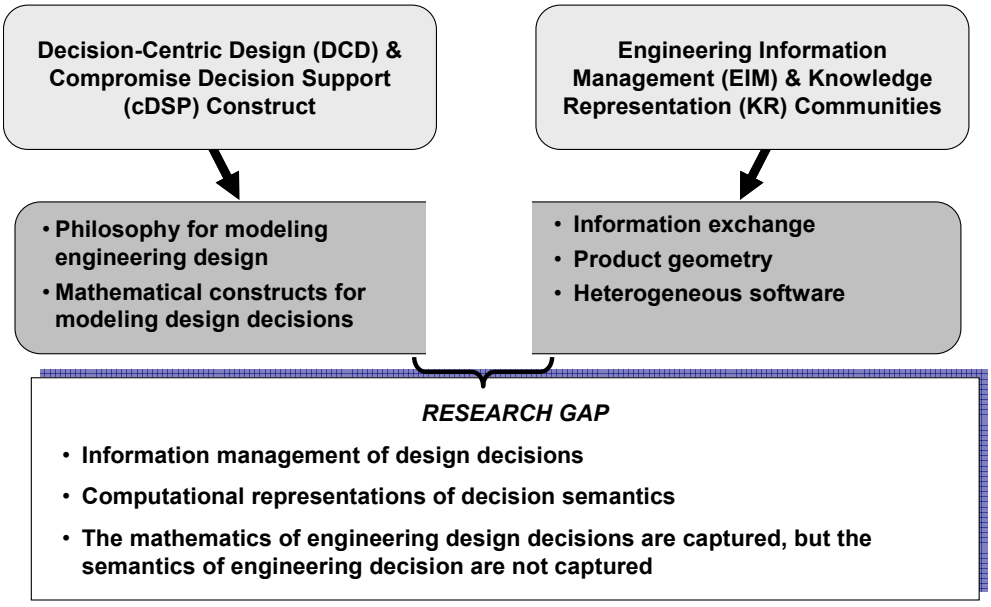


Figure 1-1: Current gaps in information management for engineering design

Thus, the overarching objective in this dissertation is to address the gaps, namely, to develop computational representations for capturing the semantics of the compromise decision support problem and the associated analysis support models. To realize the objective, an information model is developed and implemented using description logic (DL). The information model and associated implementation enable the semantics of engineering design decision to be captured and unambiguously represented and shared between multi-disciplinary designers (see Figure 1-2).

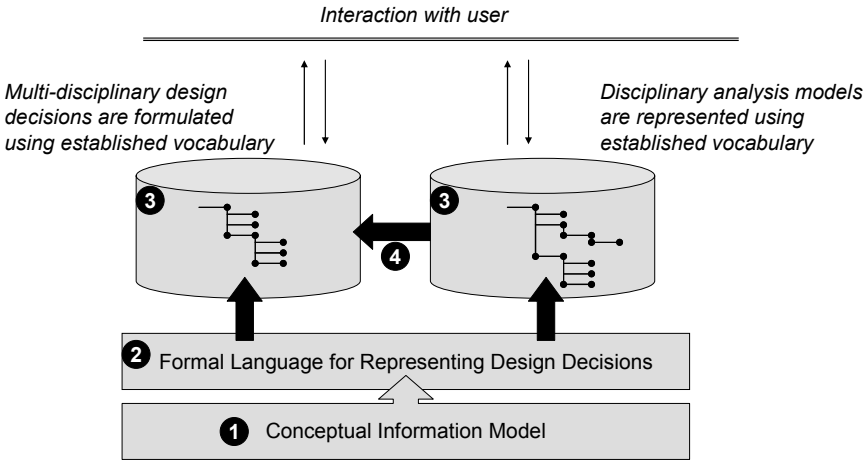


Figure 1-2: Envisioned framework for representing decision information

As illustrated in Figure 1-2, a conceptual information model (❶) is proposed as a means for unambiguously representing the information associated with the cDSP. A formal language (❷) is implemented using DL, based on the information models. Disciplinary analysis models and multi-disciplinary design decisions are formally represented using the established vocabulary and stored in a repository (❸). Decision makers are able to integrate information from multiple disciplines using the formal language as a digital interface (❹). The framework presented in Figure 1-2 is an integral

part of engineering decision support. The three components of interest in this research are:

- A **conceptual information model** for representing the domain of engineering decision making constructs and the engineering analysis support models used for decision making. The information model is an explicit representation of concepts and relationships between concepts associated with the cDSP. The conceptual information model is developed to meet specific requirements for information management of engineering decisions.
- A **DL-based representation** for engineering design decisions and engineering analysis support models in a computational means. DL representations are implemented based on the specifications of the conceptual information model. Description Logic is chosen as the representational formalism over other formalism to fulfill meta-level requirements for the model including extensibility, consistency, and organization.
- **Reasoning and querying services** that enable decision information to be captured, verified, organized, and reused. Reuse and organization of engineering information is important and a primary motivation for developing information models. Thus, standard querying and reasoning are utilized to organize and verify decision information.

The motivation for developing an information model and formal language is discussed in Section 1.1. A requirements list is developed in the context of the overarching motivation and the design problem discussed in Section 1.2. From these requirements the primary research question and hypothesis are formulated. The frame of

reference and research foundations are presented in Section 1.3. Several research gaps and opportunities are formulated based on a critical analysis of decision-centric design, the compromise decision support problem, and engineering information management. In Section 1.4, the research focus, contributions, and approach are established in light of the research foundations and overarching research objective. The primary research question and hypothesis and supporting research questions and hypotheses are formulated based on the gap analysis. The research scope and focus including the research questions, hypotheses, and contributions are discussed in Sections 1.4. Finally, the strategy for validating the hypotheses and research contributions is presented in Section 1.5.

1.1 Integrating Multiple Design Perspectives in the Product Realization Process

The increased complexity in modern products has forced a change in the way in which products are designed and developed. Engineering design is increasingly becoming a collaborative set of tasks among multi-disciplinary, distributed design teams [149]. While the advantages of multi-disciplinary, distributed design often result in increased quality and decreased product development time, limitations arise in the communication of design information across the boundaries of diverse design perspectives. Designers may create views of a product to meet their functional needs and address a particular design problem or situation. By focusing on a particular aspect of a system, a designer can abstract those characteristics and details that are important for a particular decision [163]. Decomposing a complex system into smaller sub-systems enables designers to address more manageable problems and even identify problems that are not readily apparent from a holistic perspective. However, a fundamental disadvantage of

decomposing a system into different design perspectives is the resulting information integration problems [80].

Inefficiencies in the product realization process can be attributed to many factors associated with information integration, including: (1) the lack of interoperability between heterogeneous design support tools, (2) the interactions between multiple disciplines in the design decision making process, and, (3) the inability to effectively exchange knowledge across design perspective in the context of engineering design decisions through human and computer-interpretable means. Current research has predominantly focused on addressing the interoperability between heterogeneous software. Despite the fact that multi-disciplinary design making is central in the product realization process, computational technologies have not been developed to adequately to address the issues associated with representing and integrating knowledge into decision problems. Thus, there is a need to develop computational representations to enable engineering designers to capture the information associated with design decisions. In this context, an engineering design decision is a mathematical construct of a design problem that captures the trade-off between conflicting design objectives and goals and system design parameters and variables subjected to constraints and bounds. In this dissertation design decisions are modeled as compromise decision support problems (cDSP) [25; 91; 94; 133]. Engineering design decisions are taken as basic units of communication and packages of decision-related information. Thus, the structure of engineering design decisions and the integration of multiple knowledge sources are of central importance in developing computer-interpretable formalisms.

Engineering designers formulate and subsequently make design decisions based on information from multiple sources that span various perspectives in the product realization process. In many design situations, the information used to support a design decision may be provided by geographically and temporally distributed disciplinary experts within a design-manufacturing organization (see Figure 1-3).

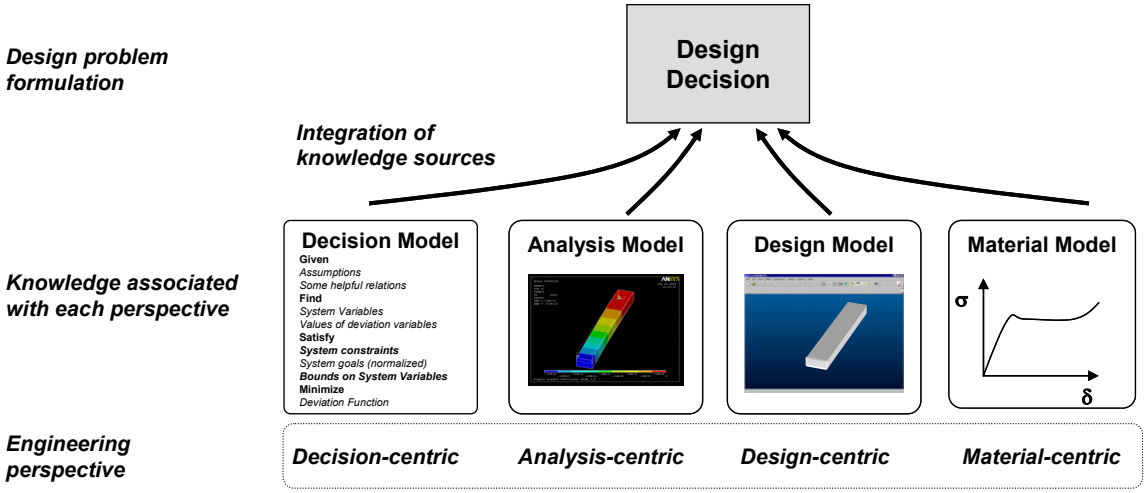


Figure 1-3: Integration of multiple models in a design decision

As illustrated in Figure 1-3, design decisions may be formulated as a cDSP using computer-aided design (CAD) model as the specification of product geometry, a material database for material properties, and finite element analysis (FEA) models to simulate the behavior of the system based on the geometry and material properties. The cDSP decision construct serves as the schema for relevant decision knowledge including information such as the designers' preferences, design information about the shape and geometry, material properties, and the behavior of the product. The crux of the problem is summarized as:

Engineering designers make decisions based on information from multiple sources that span various perspectives in the product realization process. However, the information is often independently created, limited to a single perspective, and not formally represented making it difficult to exchange and share in the context of engineering design decisions.

In this research, we are primarily interested in developing an information model and formal language for capturing the semantics of the cDSP. Current design decision modeling approaches and decision support frameworks do not address the need to capture information associated with engineering decision in a formalized, computation manner [129]. In Chapter 4, the information model and DL implementation design decision is presented. In Chapter 5, the information model is exercised for capturing the information associated with the design of structural beams and in Chapter 6 the information model is used for capturing the information associated with the design of fin array heat sinks. In Section 1.2, the challenges associated with integrating multiple design perspectives are discussed from a general perspective. A more detailed discussion and gaps associated with developing formal knowledge representation to facilitate the integration of multiple design perspectives are presented in Section 1.3.

1.2 Design Challenges and Research Opportunities

As discussed in Section 1.1, designers rely on the integration and coordination of information generated by heterogeneous design support tools from multiple design perspectives throughout the product realization process to formulate and execute engineering design decisions. Computer-based tools, technologies, and extended networks have facilitated to a limited degree the formulation and execution of

engineering design decisions in distributed, collaborative product development scenarios. In this context, these collective technologies are referred to as engineering information management (EIM) technologies.

In the broadest sense EIM technology enables stakeholders to create, share, modify, access, and view digital product representations across design extended networks. EIM tools, technologies, and standards provide the ability to share product information and knowledge across extended design teams. For example, product data management (PDM) software provides a highly structured environment to support document management and collaboration between designers. While advances in EIM have enabled distributed collaborative design, they are still inadequate for supporting the development of complex products. In this context efficiency is a measure of the swiftness with which information can be used to make a decision and effectiveness is a measure of the correctness, completeness, and comprehensiveness of the information that is used by the designer to make a decision [71]. In many cases, computer-based design support tools increase the problem by isolating information for a particular tool used in the development process [60]. Heterogeneous design support tools often rely on proprietary data formats that are difficult, if possible, to exchange between applications. ***Thus, a primary technical barrier is the integration and exchange of knowledge and information across design perspectives and domains between human designers and computer support applications.***

The problems associated with data exchange between heterogeneous software applications have not gone undetected or overlooked. There are a myriad of product models that have been developed. The product data exchange (PDE) community has put

forth significant resources and efforts focused on addressing the aforementioned data exchange problems. For example, government agencies, private companies, and universities contribute to the PDES Inc. effort to accelerate the development and acceptance of standards-based product models to enable integration and interoperability of diverse software tools. The foundation for the STEP development effort was motivated in the early international graphic exchange (IGES) standard. IGES was proposed as a means to support the exchange of product geometry data between Computer-Aided Design (CAD) systems. While the STEP effort has evolved to encompass a much wider domain than simply a CAD exchange format, the predominant focus of STEP is product geometry. The STEP standards provide a neutral mechanism through which product data can be exported in one software tools and can be imported into another through a standardized "language". However, STEP and many other standardized product models primarily focus on capturing and exchanging geometric design specifications [4; 162]. While geometric design information constitutes a significant amount of product information, additional information must be captured to enable computer-based product realization. Sharing product geometry and CAD models across distributed networks is not adequate to fully support computer-based product realization processes [147].

Next generation product models and EIM technologies must increase the breadth of information captured beyond purely geometric information [146]. As a result, several product modeling efforts have been proposed with a focus on capturing increased product information such as product behavior, function, and rationale to name a few. For example, researchers in the Manufacturing Engineering Laboratory at NIST have proposed a family of product models and their interactions that capture life-cycle

information [43-46]. For example, the core product model (CPM) serves as a conceptual architecture the coordinates a master product model and several aspect view models to address the multi-disciplinary nature of engineering product realization (see Figure 1-4).

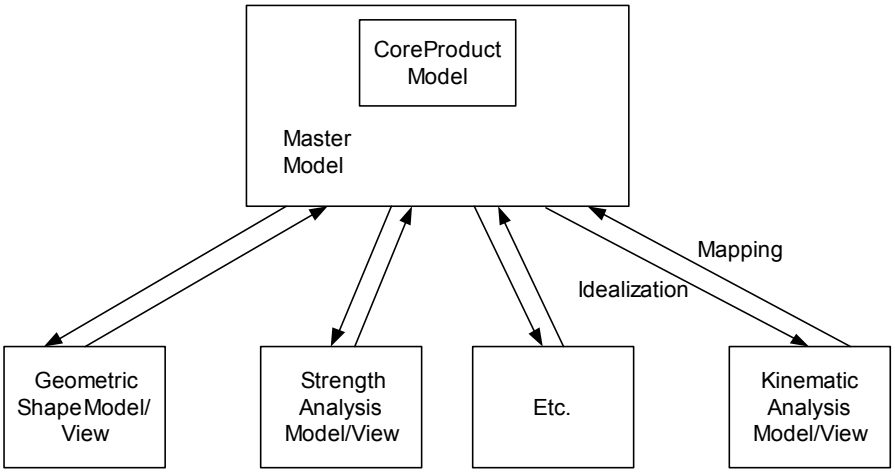


Figure 1-4: Conceptual architecture for associating multiple domains [46]

One such research effort that builds on the conceptual architecture illustrated in Figure 1-7, is the integration of engineering design and analysis domains. In general, design-analysis integration (DAI) addresses the seamless integration between design and analysis by linking computer-based design and analysis models [114]. Several similar architectures and product models have been proposed for addressing problems in design analysis integration [7; 40; 45; 63-65; 114; 169]. However, the primary focus of DAI research has remained on developing richer representation for sharing design and analysis geometry. ***A fundamental shortcoming in current DAI research and product models technology development is the inability to capture the decision-related knowledge and the context in which the "linkages" between design and analysis are established.***

Decision-centric design is proposed as an approach for modeling the product realization process as a set of inter-related design decisions. In decision-centric design,

domain-independent decision constructs provide a common "language" for modeling design problems that require the integration and composition of knowledge from multi-disciplines and multiple design perspectives [94]. General framework have been proposed to support collaborative, distributed decision making in engineering design [47; 129]. However, the current decision frameworks provide an environment for representing engineering design decisions in terms of mathematics, but do not enable designers to systematically capture the semantics of knowledge associated with design decisions. This is in part, due to the lack of formalization of decision related knowledge and support models.

The compromise DSP, a specific type of decision construct, is a multi-objective decision model that is a hybrid formulation based on Mathematical Programming and Goal Programming [91], a detailed discussion of multi-objective decision making and the compromise decision support problem is presented in Section 2.3. The cDSP is used to determine the values of design variables that satisfy a set of constraints while achieving a set of conflicting goals as closely as possible. However, a formal language to represent engineering design decision has not been developed. While there has been a handful of information models proposed as a means for capturing the decision knowledge, they are focused on the persistent storage of databases for engineering design decisions [25; 69; 71]. Additionally, the representation have not kept pace with current research development in knowledge management and representations such as ontologies [60]. However, the current approaches are *data-oriented* and provide a means for modeling mathematical decision knowledge. ***While it is essential to capture the mathematical formulation of a design decision in a computational means, the semantics and the***

integration of knowledge from multiple design perspectives of the decision are often lost. Thus, there is a need to develop computer-interpretable representations that capture the semantics of design decisions and enable efficient integration of knowledge from multiple sources.

Gruber [60] proposes the notion of "a computational environment in which explicitly represented knowledge serves as a communication medium among people and their programs." In this context, Gruber offers that formal knowledge representations enable collaboration and communication amongst stakeholders in the product realization process. Advances in knowledge representation and ontologies have provided a viable foundation for capturing the semantics of design decisions. However, the DBD and MDO communities have not committed to developed advances representations for information management. ***Thus, there is a need to develop formal knowledge representation language of engineering design decisions to enable designers to capture and utilize knowledge from multiple sources.***

The challenges associated with integrating multiple perspectives summarized in Table 1-1 are far too complex to address in this dissertation. However, they provide a sense of the immensity of the problems in multi-disciplinary decision making.

Table 1-1: Challenges associated with integrating multiple design perspectives

- A primary technical barrier is the integration and exchange of knowledge and information across design perspectives and domains between human designers and computer support applications.
- A fundamental shortcoming in current DAI research and product models technology development is the inability to capture the decision-related knowledge and the context in which the "linkages" between design and analysis are established.
- While it is essential to capture the mathematical formulation of a design decision in a computational means, the semantics and the integration of knowledge from multiple design perspectives of the decision are often lost. Thus, there is a need to develop computer-interpretable representations that capture the semantics of design decisions and enable efficient integration of knowledge from multiple sources.
- There is a need to develop formal knowledge representation language of engineering design decisions to enable designers to capture and utilize knowledge from multiple sources.

The challenges are refined and the scope is limited to reflect the focus of this dissertation as:

- Capturing the semantics of engineering design decisions by developing explicit representations of the structure of decision information
- The development of a computational language for communication decision-related information between designers

The conceptual information model and computational representation developed in this research is motivated by the challenges summarized in Table 1-1 in the context of distributed, collaborative design decision making. The principal challenge is to enable the integration and communication of information from multiple design disciplines in engineering design decisions. Given the design challenges and the primary objective in this research, the requirements associated with realizing the computational framework is identified. A summary list of requirements for capturing decision-related knowledge in the product realization process is included in Table 1-2.

Table 1-2: Decision-centric design information modeling requirements

- Provide a structured means for capturing engineering design information
- Capture the semantics associated with engineering design decision
- Facilitate the integration of information from multiple sources for decision making
- Support changes and additions of new design and analysis information in an extensible manner
- Provide a computer-interpretable means for representing decision-related information
- Provide algorithms for retrieving and organizing decision-related information

In light of the research challenges and requirements, the primary research question addressed in this dissertation is:

Primary Research Question*: How can information from multiple sources be (a) systematically captured and (b) formally represented in a computational means to facilitate integration in decision-centric design?*

To address the afore-mentioned challenges, an information model and computational representation are developed in this dissertation. The frame of reference and research foundations are presented in Section 1.3, followed by a critical review and discussion.

1.3 Frame of Reference and Research Foundation

The fundamental building blocks for developing the information model include the decision-centric design (DCD) philosophy and compromise decision support problem (cDSP) and description logic (DL) representations. Each of these foundational building blocks is discussed on the context of developing a formal language to support design decision making. Limitations, gaps, and research opportunities are identified in the current state of art that provide motivation for the research questions and hypotheses addressed in this research. The research foundations are presented in light of conceptual-level research questions and implementation level-research questions. The primary research question, presented in the previous section is discussed in light of the overarching research objective as the following:

Research Objective*: Develop a computational knowledge representation (i.e., a formal language) for capturing the semantics of engineering design decisions in a structured manner to facilitate the exchange and integration of knowledge from multiple design perspectives throughout the product realization process. The representation should enable multi-objective decision to be formulated in an extensible and robust manner.*

The foundational building blocks for realizing the primary objective are critically analyzed and research questions and objectives are developed.

1.3.1 Decision-Centric Design and the Decision Support Problem Technique

The challenges associated with integrating multiple design perspectives are addressed in this dissertation through a decision-centric design (DCD) approach. Panchal and co-authors [108] propose DCD as an augmentation and extension to decision-based design (DBD). DBD, originally proposed by Mistree and co-authors [94], is a conceptualization of the product realization process in the context of domain-independent decision constructs. These domain-independent decision constructs provide a common platform, or "language", for modeling design problems that require the integration and composition of knowledge from multi-disciplines and multiple design perspectives [94]. Thus, stakeholders are able to collaborate throughout the product realization process at decision points.

In DBD, the primary role of a designer is to make decisions throughout the realization process. A core extension to DBD is the focus on design tasks and activities that generate information to support design decisions. Thus, design decisions and support tasks are of central importance in modeling the design process. The principle point abstracted from both DBD and DCD is that *the engineering product realization process is represented as a set of inter-related design decision and supporting activities which serves as units of communication and collaboration between stakeholders in the design process.* Decision-centric design facilitates the integration of multiple design perspectives by establishing a common construct through which designers from various perspectives can integrate and reconcile knowledge related to a particular design problem.

Mistree and co-authors state that DBD is a different perspective on which to develop methods to support design. In DBD the principal role of a designer is to make decisions.

Thus, design processes can be modeled through decisions that serve as markers for progression from design initiation to completion and as a unit of communication and support tasks that create, manipulate, and modify information to support design decisions. Designing a product becomes an issue of information processing in which information about the needs of the product are converted into knowledge about the product. A decision is an "irrevocable allocation of resources" that has two important characteristics: (1) a decision is made at an instant in time and (2) a decision must be made with the information available at the time it is made. The principle characteristics of DBD include:

- The primary role of the designers is to make decisions
- Designing a product involved decisions that may be executed sequentially and/or in concurrently
- Design involved hierarchical decision making and the interaction between decisions must be taken into account

The Decision Support Problem (DSP) Technique [28; 90; 93-95; 99] is a specific implementation of DBD for modeling engineering design decisions. The DSP Technique is rooted in systems thinking, and approach for formulating DSPs in engineering design. The decisions commonly encountered by designers in a product realization process are characterized in Table 1-3.

Table 1-3: Characteristics of decisions encountered in design

- *Multileveled* – Systems are composed of subsystems and so on. Different systems are subsystems for other systems. We cannot model real-world engineering systems. We cannot establish the model to represent the system in all ways. We cannot represent all decisions that are made in the conception and over the entire life cycle of the system. We can simply create models on higher levels.
- *Multidimensional* – A system may be composed of subsystems (recursive definition of subsystems – hierarchical). Additionally the decision made in the design process can be concurrent or sequential in nature.
- *Multidisciplinary* – Engineering is based on information from different disciplines. Decisions are based on information from multiple disciplines – we take this for granted in everyday life and engineering decisions.
- *Satisficing solutions* – Real world optimization is impossible. Real world optimization is impossible – engineers are satisficers who accept "good-enough" alternatives.

The DSP Technique is not focused on *fully automating* design, but rather on providing support to human decision makers. The DSP Technique enables designers to model the product realization process from a neutral, decision-based perspective – namely through the language of design decisions. The DSP Technique consists of two phases: meta-design and design. The first phase is a "meta-design" phase where the necessary decisions in the design process of a product are identified and structured based on several axioms for modeling engineering design decisions [95]. In meta-design, the problem is partitioned and design problems are identified in terms of decision problems.

In the second phase, the domain dependent information is formulated and structured in accordance with established decision constructs.

Foundational to the DSP Technique are Decision Support Problems (DSP) [18; 94]. DSPs are mathematical constructs for modeling engineering design decisions. DSPs provide the structure and organization of relevant decision-making information to model decisions commonly encountered in a product realization process. The formulation and solution to DSPs enable designers to model several types of decisions typically encountered in design. Mistree and co-authors [94] assert that there are two primary types of decisions: the selection decision support problem [18; 49; 92] and the compromise decision support problem[91]. Coupled decisions [18] and that all other types of decisions can be derived from these basic decision problems (see Figure 1-5).

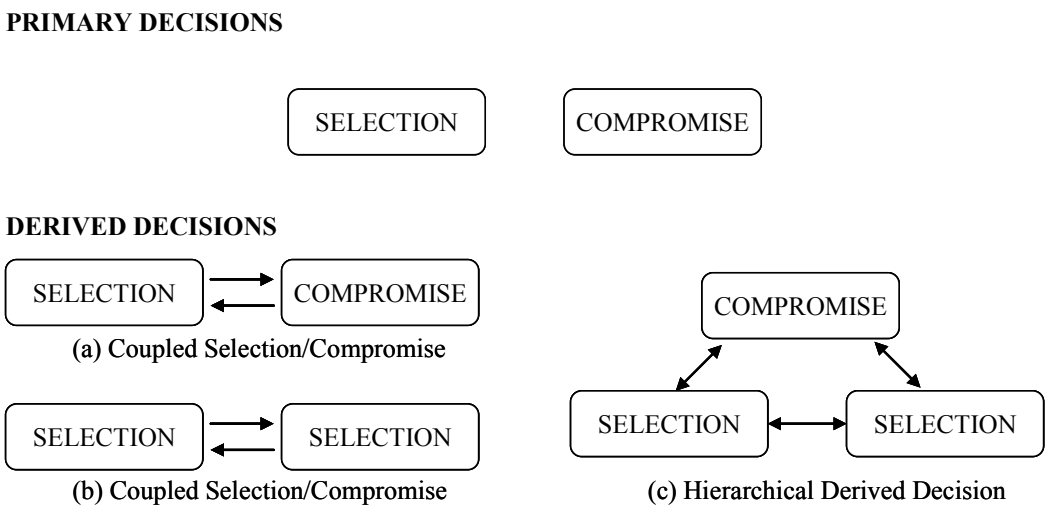


Figure 1-5: Primary and derived decisions [95]

DCD provides a philosophical foundation on which to model the engineering design process. Mistree and co-authors assert that DCD enables design to be modeled from a discipline independent approach and provide a common "language" through which

designers can communicate and collaborate. However, the representation and subsequent integration of decision-making knowledge in a computational means is limited. A first step associated with realizing a DCD view of design is the development of computer-based decision constructs. Decision support problems (DSPs) provide a means for modeling design decision and provide the mathematical-based formulation in the form of decision templates. However, the representation and integration of information and knowledge within engineering design decisions are not adequately captured. Therefore, the following attention directing question is posed:

- *What information should be captured about engineering design decision and associated support models?*
- *How can decision-centric design constructs be formally represented in a computer-interpretable means to facilitate the integration of multiple design perspectives in the product realization process?*
- *What representational formalism meets the requirement for modeling in engineering decision making?*

1.3.2 The Compromise Decision Support Problem

Several mathematical formulations have been proposed for modeling decisions in engineering design. The compromise DSP (cDSP) is a formulation for multi-objective decision problems encountered in design, as characterized in Table 1-3. The cDSP is a generic class of multi-objective decision problems that are common to engineering design situations [81]. The cDSP is a multi-objective decision model that is a hybrid formulation based on Mathematical Programming and Goal Programming [25; 91; 94; 133]. The cDSP is a domain independent construct that enables designers to model decision to

determine the "right" values (or combination) of design variables (e.g., system parameters), such that, the system is feasible with respect to constraints, preferences, bounds, and goals, and that system performance is maximized The cDSP is used to model trade-off decisions in design. The cDSP can be used as a means for packaging relevant decision-making information and sharing it with stakeholders in the product realization process. The compromise DSP has been successfully used in designing aircraft, thermal energy systems, mechanisms, structural systems, ships, material composite design, aircraft engines, satellite trajectories, and design for manufacture.

The cDSP has two representations; a word formulation that represents the design problem conceptually and the mathematical formulation that can be solved using traditional optimization techniques. The word formulation of the cDSP provides a means for capturing knowledge about the design problem as a high-level template. The word formulation of the cDSP must be transformed into a corresponding mathematical form to facilitate solving the design problem. There is a one-to-one mapping between the word formulation and mathematical form (see Figure 1-6).

The word formulation of the cDSP is a *pseudo code-like* representation of engineering design problems based on established keywords and descriptors. The word formulation enables a designer to model the design problem as a decision support problem while minimizing the need to *implement* the design decision in terms of mathematics and/or a particular programming language. The mathematical formulation requires translating the conceptual design problem to a computer program to be solved using traditional optimization routines.

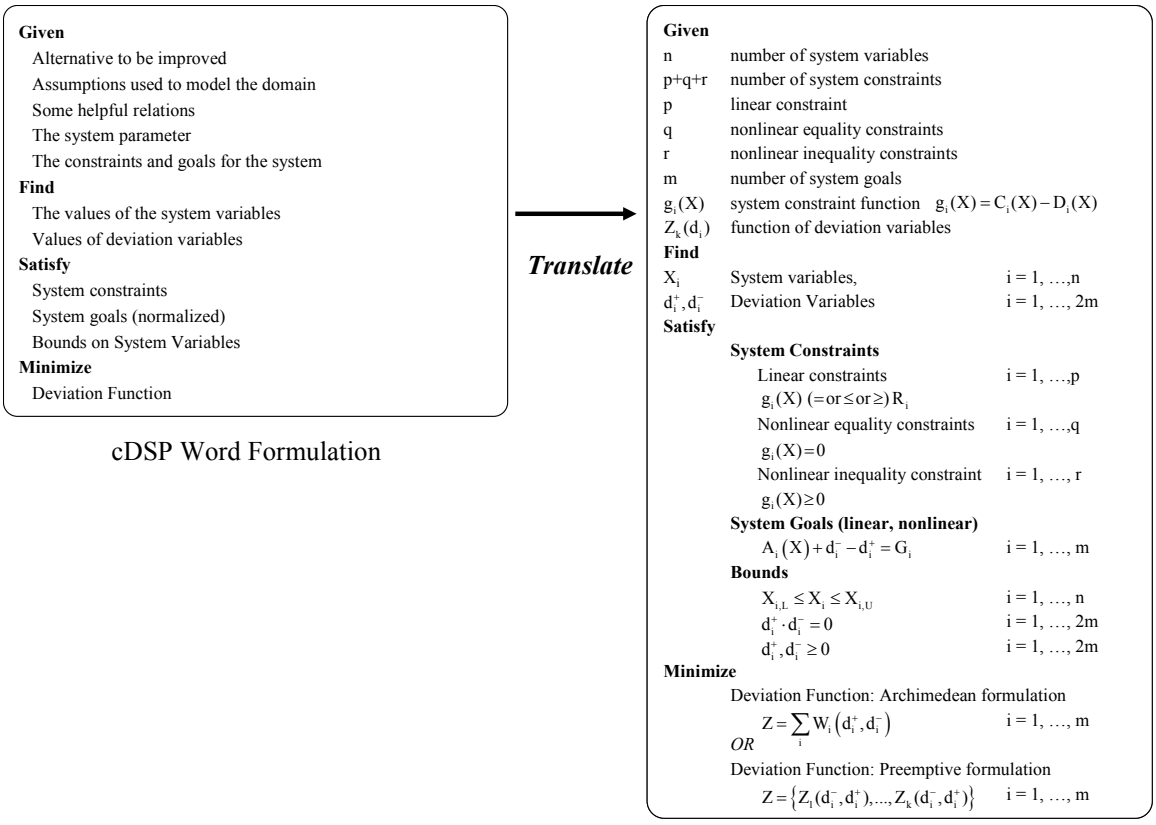


Figure 1-6: Association between word formulation and mathematical formulation of cDSP [91]

The cDSP construct provides high-level guidance for modeling design problems as *multi-disciplinary design problem* and serves as a "package" of relevant decision making knowledge in terms of the optimization problem. However the semantics associated with the integration of knowledge and subsequent decision formulation are often lost or not fully captured within the cDSP. Several attention directing questions are posed that bring these issues to light:

- *What is the structure of information associated with design decisions?*
- *What are the sources of the constraints, are they imposed by the designer, the*

analysts?

- *What restrictions and assumptions are encapsulated in analysis models?*
- *What responses are of interest in the design decision?*
- *What models can be utilized to support the objectives given certain design information?*
- *How can decision be structured to facilitate reuse?*
- *How should the decision information be captured?*
- *How and what disciplinary analysis information is captured?*
- *What computational representations provide support for engineering problems?*
- *How can the decision information be reused and/or organized?*
- *How is decision-related information shared?*

Additionally, it is argued that the DSP Technique and the cDSP provide a common language through which designers can communicate and collaborate. However, the degree to which computer-interpretable representations of engineering design decisions have been formalized is limited. Thus, a primary shortcoming in multi-objective decision making can be attributed to the lack of formal computer-interpretable models for capturing the knowledge associated with design decisions. Additionally, the cDSP construct provides designers with a means for representing design decisions through an established structure. However, the cDSP does not support the need to capture and represent the information intensive nature of engineering design perspectives and the associated computer-based product models. The cDSP captures decision-related information in flat structure. In other words, the complex relationships between system variables, parameters, constraints, and models are not persistent when the cDSP is

represented in mathematical form. The integration of knowledge in a cDSP is illustrated graphically in Figure 1-7 for the design of a cantilever beam.

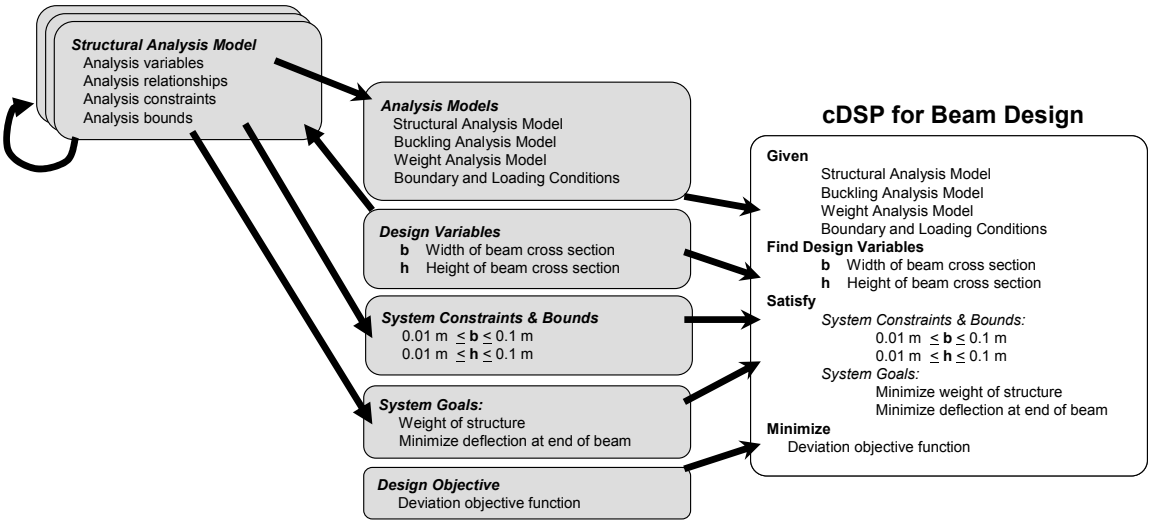


Figure 1-7: Anatomy of information associated with a cDSP

As illustrated in Figure 1-7, the cDSP may be comprised of knowledge from multiple analysis models and design models, each with a set of constraints, bounds, and relationships. In order to realize the integration and exchange of knowledge in a computational environment, a common knowledge representation is needed. Despite advances in engineering information modeling, formal representations of engineering design decisions have not been developed. Thus, the following research question is posed:

***Research Question 1a**: How can the structure and semantics of the compromise decision support problem be captured?*

The focus of Research Question 1 addresses the need to develop an information model of engineering design decisions. Information models are explicit representations of the objects and relationships between objects in a particular domain. Information models

enable agents to communicate through an agreed upon terminology. The problem of knowledge sharing and information exchange in engineering design is by no means a new concept. In fact information management technologies have been used extensively to manage business process applications. However, the need to represent *decision making* knowledge has been grossly overlooked, ultimately leaving a large gap in collaborative product realization. Thus, formal decision making models are based on existing technologies for capturing knowledge and information in engineering.

1.3.3 Integration of Disciplinary Analysis Models

As illustrated in Figure 1-7, the information in a cDSP is based on the integration of knowledge from several models that may span multiple design perspectives. For example in the cDSP several analysis models may be used to simulate the behavior of a product for multiple objectives. The knowledge captured in each of the disciplinary analysis models must be integrated and coordinated with other models in a unified design decision. Design-analysis integration (DAI) is the area of research that addresses the need to integrate engineering design and analysis models in a computational means. Design-analysis integration (DAI) is defined as the seamless integration between design and analysis perspectives by capturing the relationships between computer-based design and analysis models. The "linkages" between design and analysis space enable designers to capture many different types of relationships, including heuristic and mathematical relationships, between engineering design models and engineering analysis models. A common type of relationship captured are geometric and phenomenological idealizations and simplifications [50]. Peak and co-authors [113] address parameter-level *DAI* by capturing fine-grained associativities between traditional design (CAD) and analysis

(CAE) tools through the multi-representation architecture (MRA) [112]. The MRA is realized through the development of the constrained objects (COBs) knowledge representation language [165]. However, the integration of design and analysis has predominantly been addressed independent of design decision making. While research and development efforts in the domain of DAI have addressed the need to exchange product-related data, they have not adequately captured the decision making context. As a result, the information models and focus of design analysis integration have been limited to geometric information. The current information models and representation do not adequately capture information to support multi-objective design decisions. Thus, the second part of research question one is:

***Research Question 1b**: How can the structure and semantics of analysis support models be captured to facilitate integration in the cDSP?*

Answering Research Question 1b addresses the need to develop a conceptual information model for representing analysis information. Research question 1a and 1b are closely related and address the need to develop information representation of decision-related information. However, a key requirement is information representations be computer interpretable. In the following sections, several foundational building blocks are presented as a means for developing computer-interpretable decision models.

1.3.4 Information Modeling in Engineering Design

The need for computer-based models to support engineering design is fueled by the immense amount of digital information generated about the product throughout the realization process and the dependence on computational tools to support design. Fulton

and Chadha [53] assert information systems are essential to manage and integrate product data generated throughout the product realization process. Information is considered to be a corporate asset and thus security, consistency, and availability are of primary importance. Not surprisingly, design support tools and technologies have matured at a rapid rate and have permeated many aspects and domains of product development.

Standardized information models for capturing product information, such as STEP, have enabled product-related design information to be shared amongst heterogeneous design tools to some degree [57; 65]. STEP has significantly impacted, both technically and economically, the design of complex engineering systems by providing increased information exchange and reducing the effort and cost of interoperability [4; 54]. Several researchers have proposed models for capturing product and process information [5; 144]. Similarly, Panchal and colleagues [108; 109] propose eXtensible Markup Language (XML) templates for capturing decision-centric design (DCD) activities, tasks, and information transformations in the design process to support the engineering design decisions. Recently, models have been proposed for product data management (PDM) and product lifecycle management (PLM) for integrating information from a systems-level perspective [43-45]. Similar information models have been proposed for integrating design and analysis domains [112]. Design repositories have been developed for analysis information [59; 97] and function-based design information [23; 24; 74; 75]. However, STEP and many other standardized product models primarily focus on capturing and exchanging geometric design specifications. While geometric design information constitutes a significant amount of product information, additional information must be

captured to enable computer-based product realization. Thus, the second research question is posed:

Research Question 2: *How can the information associated with the cDSP and analysis support models are represented in a computational environment?*

An underlying motivation for developing computational representation of engineering decision knowledge is to facilitate reuse and retrieval of design information. Information reuse has been addressed by several researchers including the development of design repositories, software and component reuse in design [158; 159], simulation reuse and reconfiguration [82; 110; 124], process modeling reuse [109] and reuse for reconfiguration of optimization problems [10; 11]. However, the reuse and retrieval of engineering design decisions has not been addressed. Salas and Townsend identify several requirements pertaining to the retrieval and reuse of information for decision formulation including (1) access to database management feature and (2) retrieval of decision information. Thus, a central challenge addressed in this dissertation is how to retrieve and reuse information for engineering design decision. The following research question is posed:

Research Question 3: *How can cDSP-related information be organized and retrieved to enable reuse?*

In this dissertation, Description Logic provides a formal representation for representing information in a computational means and supports reasoning and organization of information.

1.4 Research Scope & Focus, Approach, and Contributions

The motivation, research challenges, and research foundations are discussed in Sections 1.1 through 1.3. The primary focus, contributions, and approach for the research are presented in this section. The focus in this research is on establishing an information model and computational representation to enable the semantics of engineering design decision information to be captured. The representation is proposed to support the integration of information from multiple sources in the product realization process. In Section 1.4.1, the research questions and hypotheses are established in the context of the challenges and requirements identified in Section 1.2. In Section 1.4.2, research contributions are summarized at the conceptual and implementation levels. In Section 1.5, a strategy for validating the research hypotheses is presented.

1.4.1 Fundamental Research Questions and Hypotheses

Based on the previous discussion and review, the primary problem addressed is summarized as follows:

Engineering designers make decisions based on information from multiple sources that span various perspectives in the product realization process. However, the information is often independent, limited to a single perspective, and not formally represented making it difficult to exchange and share in the context of engineering design decisions.

In light of the primary problem statement, the principle goal in this dissertation is

Develop a computational knowledge representation (i.e., a formal language) for capturing the semantics of engineering design decisions to facilitate the integration of

knowledge from multiple design perspectives.

The requirement for addressing the challenges associated with integrating multiple design perspectives in the context of decision centric design are mapped to the three research questions in Table 1-4.

Table 1-4: Mapping the requirement to research questions

Requirements	Research Questions
• Provide a structured means for capturing engineering design information • Capture the semantics associated with engineering design decision • Facilitate the integration of information from multiple sources for decision making	**RQ1a:** How can the structure and semantics of the compromise decision support problem be captured? **RQ1b:** How can the structure and semantics of analysis support models be captured to facilitate integration in the cDSP?
• Support changes and additions of new design and analysis information in an extensible manner	**RQ2:** How can the information and knowledge associated with cDSP and analysis support models are represented in a computational environment?
• Provide a computer-interpretable means for representing decision-related information • Provide algorithms for retrieving and organizing decision-related information	**RQ3:** How can cDSP-related information be organized and retrieved to enable reuse?

The focus in completing this research is to develop an information model for explicitly representing and modeling engineering design decisions. The information model is established based on decision-centric design principles and information modeling formalisms. Specifically, the information associated with the compromise decision support problem (cDSP) and support models is formalized using description

logic (DL). The notional architecture for realizing the computational framework is illustrated in Figure 1-8.

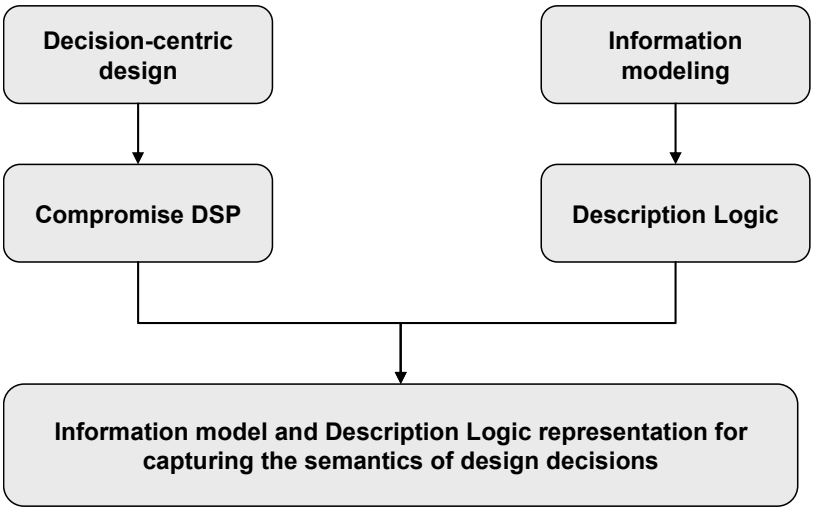


Figure 1-8: Integration of formal knowledge representations with decision-centric design foundations to establish a computational framework

As shown in Table 1-5, the primary and supporting research hypotheses are correlated directly to sub-research questions in the context of developing a computational framework to formally capture knowledge associated with design decisions.

The purpose of both Hypotheses 1 and 2 is to develop an information model for explicitly capturing engineering design decisions. First, information modeling is concerned with developing computer-based symbols for modeling a particular domain of discourse [100]. Second, computer-based representations for explicitly capturing knowledge associated with decision constructs and analysis models in decision-centric design, thus enable communication between decision stakeholders in distributed, collaborative decision making. In Hypothesis 1, the information associated with the

compromise decision support problem (cDSP) is formalized to facilitate the realization of a computational framework for representing engineering decisions.

Table 1-5: Research questions and hypotheses

Research Question	Research Hypothesis
Primary Research Question: How can information from multiple sources be (a) systematically captured and (b) formally represented in a computational means to facilitate integration in decision-centric design?	**Primary Hypothesis:** Description Logic-based information models will provide a formal language for integrating multi-disciplinary decision knowledge.
RQ1a: How can the structure and semantics of the compromise decision support problem be captured? **RQ1b**: How can the structure and semantics of analysis support models be captured to facilitate integration in the cDSP?	**Hypothesis 1**: Information models can be developed to explicitly represent the concepts, and properties associated with cDSPs and analysis support models
RQ2: How can the information associated with the cDSP and analysis support models are represented in a computational environment?	**Hypothesis 2:** Description Logic can be used for realizing the information models in a computational means
RQ3: How can cDSP-related information be organized and retrieved to enable reuse?	**Hypothesis 3**: Reasoning and querying services supported by Description Logic can be utilized for organizing and retrieving decision related information

In Hypothesis 2, the focus is on representing engineering analysis knowledge to enable the integration in engineering decisions. Hypothesis 1 and 2 are closely related and address the need for computer-based representations for establishing ontologies as a common language for decision-centric design. Hypothesis 3 builds on the ontologies

established in Hypothesis 1 and 2. In Hypothesis 3, knowledge reasoning and querying services are leveraged to enable the search and retrieval of decision-related knowledge.

In Hypothesis 1 (H1), it is asserted that an information model can be developed to explicitly represent the information associated with engineering design decisions. As previously discussed, the utilization of engineering information management technology and knowledge representations for capturing decision related information is limited. As suggested in Hypothesis 1, an information model is proposed in this dissertation for explicitly capturing and structuring the information associated with multi-disciplinary design decisions.

Hypothesis 2 (H2) builds on information model developed in Hypothesis 1 (H1). In H2 the focus is on developing computer-interpretable representations of decision information. Particularly in this research, Description Logic is utilized as the formalism for modeling decisions. The DL ontology is proposed as a means for sharing a common understanding and structure of information between stakeholders in the decision-making process, to enable the reuse of knowledge across design perspectives, and to establish a declarative representation of decision making knowledge.

In Hypothesis 3 (H3), reasoning and querying services are proposed as a means for reusing and organizing decision and analysis related knowledge. Reasoning and querying services are performed on the knowledge representations from Hypothesis 2to facilitate decision information reuse. Standard reasoning services supported by Description Logics provide the ability to organize decision related information in a hierarchical fashion and retrieve information from the knowledge base for reuse.

1.4.2 Research Contributions

The overarching objective in this research is to address the gaps identified in Section 1.2 in the context of the research foundations. The research contributions in this dissertation are established by testing the research hypotheses introduced in the previous section. The overarching research contributions in this dissertation correspond to the primary research question and hypothesis (summarized in Figure 1-9).

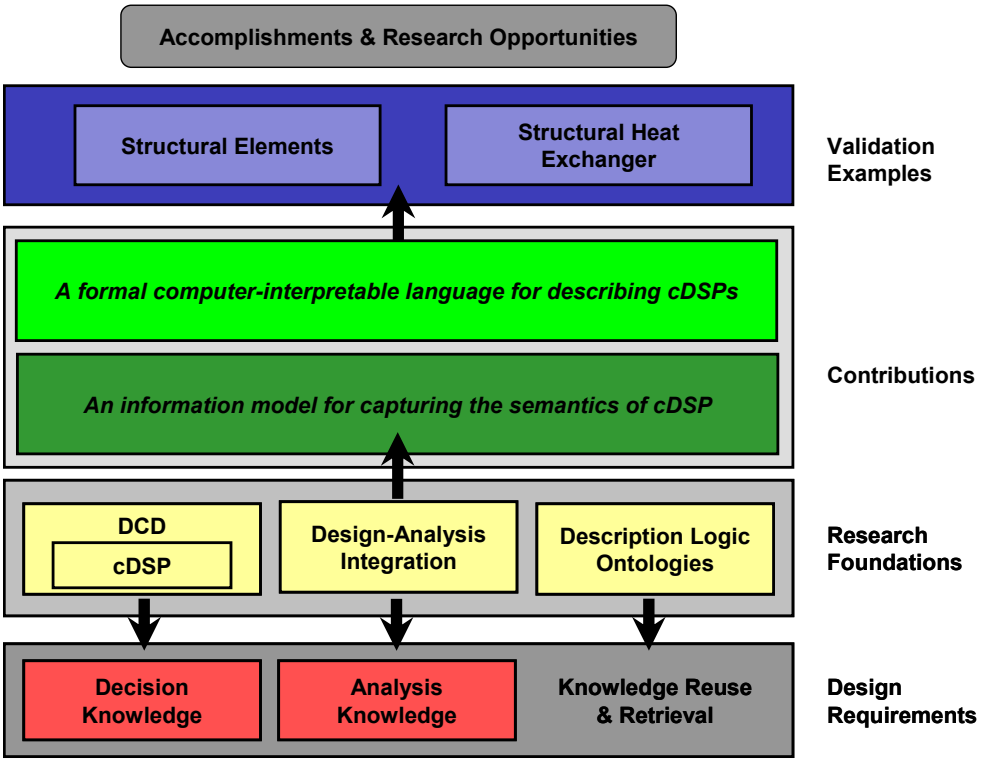


Figure 1-9: Summary of requirements, foundations, contributions, and examples

Contributions from this dissertation are summarized as follows:

- A systematic method for formulating multi-objective engineering design decisions. The method is comprised of seven phases that are divided into two sub-methods. The first sub-method is focused on capturing decision related information and the second

sub-method is focused on characterizing disciplinary analysis models for reuse. The method provides a structure framework for capturing decision related knowledge.

- An information model for capturing the semantics of engineering design decisions. The information enables decision related information to be captured in a structured manner in accordance with the systematic design method. The information models serves as the foundation for developing computational implementations.
- A DL-based computer-processible representation of engineering decision information is proposed. The DL implementation enables designers to create descriptions of design problems using an established vocabulary and predetermined set of logical constructs.. The knowledge representation serves as a "common language" through which designers from multiple perspectives can integrate knowledge and formulate design decisions.
- A knowledge representation for explicitly representing assumptions, limitations, and constraints on analysis models. In many cases, analysis models may be used to support design decisions without regard to the limitations and assumptions of the model, thus resulting in invalid design decisions. Several examples are developed to illustrate the importance of capturing the limitations and assumption analysis models in design decision making
- An application of DL in engineering design, specifically engineering design making. Description Logic has predominantly been researched in the fields of computer science and medical information management. However, DL has not been extensively applied for engineering information management (although it is increasing). A critical

assessment of DL capabilities correlated to engineering design problem requirements is completed.

The contributions in this research are validated and verified in the context of the research questions and hypotheses. The approach for validating and verifying the contributions is presented in the following section.

1.5 Verification and Validation of Contributions in this Research

Validation and verification of the information model and DL-implementation developed in this research is based on the validation square strategy proposed by Pederson and co-authors [117]. In this context, validation refers to the internal consistency and verification deals with the justification of the knowledge claims associated with the proposed representation. In other words, validation addresses the question "Did we develop the method correctly?" and verification answers the question "Did we develop the correct method?"

In many applications validation and verification of engineering research can be completed as a component of scientific inquiry based on logical induction and/or deduction. Traditional engineering research based on mathematical modeling can be validated and verified using this approach. However, the field of design theory and method development is often based on subjective, qualitative statements and mathematical models which make validation and verification of design methodologies difficult. Pederson and co-authors pose the question directly related to this dissertation, namely: How can design research and design methods be validated? Thus, this

dissertation is validated and verified based on the approach / strategy developed by Pederson and co-authors.

1.5.1 The Validation Square

Pederson and co-authors [117] propose an alternative means for validation and verification that is geared towards design methodologies based on the roots of epistemology. In this context validation of engineering research is defined as: "a process of building confidence in its usefulness with respect to a purpose." For design methods, usefulness is whether the method results in a solution correctly and whether the method provides the correct solutions. The validation square strategy is based on both qualitative and quantitative measures of the design method being verified and validated. This process of validation is embodied in the Validation Square (see Figure 1-10).

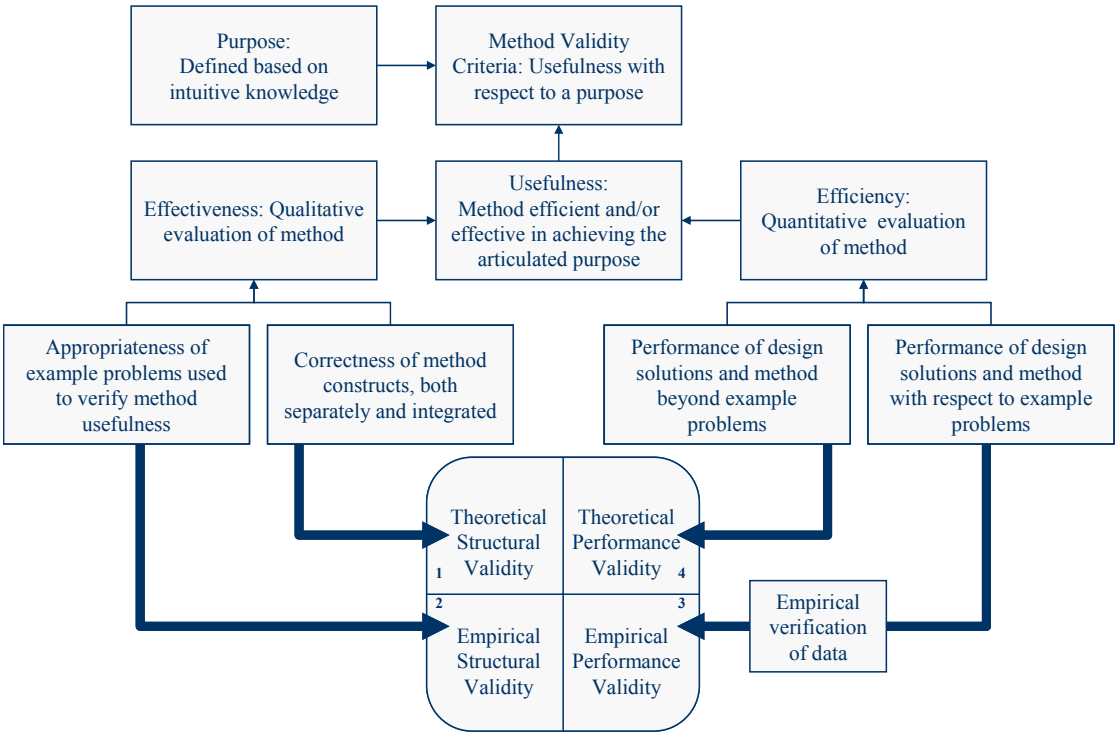


Figure 1-10: The Validation Square approach [117]

Structural validation (quadrants 1 and 2 in Figure 1-10) is a **qualitative** process consisting of three primary activities (1) accepting the individual constructs in the method; (2) accepting the internal consistency or the way the individual constructs are put together in the method; and accepting the appropriateness of the example problems that will be used to verify the performance of the method. Specifically, **theoretical structural validity** (quadrant 1) is focused on accepting the individual components and constructs comprising the method and the internal consistency of the method from an integrated viewpoint. **Empirical structural validity** (quadrant 2) involves building confidence in the appropriateness of the example problems developed to test the various aspects and claims about the method. The degree to which theoretical structural validity is completed is directly related to the scale and scope of the chosen example problems.

Performance validation (quadrants 3 and 4 in Figure 1-10) is a quantitative process that involves (1) accepting that the outcome of the method is useful with respect to the chosen example problems; (2) accepting the usefulness of the method through the application of the example problems; and (3) accepting the usefulness of the method beyond the proposed examples. Empirical performance validity (quadrant 3) is completed in the context of example problems. The objective of empirical performance validity is to build confidence in the usefulness of a method using example problems that address the aspects of the design method comprehensively. Theoretical performance validity (quadrant 4) is aimed at building confidence in the generality of the method and accepting that the method is useful beyond the example problems. Pederson and co-authors state that through analytic generalization and theory development, a "leap of faith" from empirical performance validity to theoretical performance validity must be

taken. This jump is taken to show the method is useful in a more general sense. The size of the jump is related to the example problems that are used as a test-bed for the method.

1.5.2 Verification and Validation in this Dissertation

As previously stated, the Validation Square strategy proposed by Pederson and co-authors [117] for verifying and validating engineering design method is used as the framework in this dissertation. A general approach for verification and validation of the contributions in this dissertation is summarized in Table 1-6. Specific aspects of validation and verification of the information and DL representation is discussed throughout the dissertation (see Table 1-7).

Table 1-6: General strategy for validation and verification of engineering design methods based on the validation square [117]

Validation Square Quadrant	Approach for Validation and Verification
Theoretical structural validity	• Critically evaluating relevant publications and current state of the art. • Conducting a gap analysis of the relevant constructs and existing computational frameworks and knowledge representations. • Establishing criteria and requirements to check the information model and knowledge representation. • Internal consistency of the DL representations.

Table 1 – 6: General strategy for validation and verification of engineering design methods based on the validation square [117] (continued)

Empirical structural validity	• Development and documentation of example problems through which the information model is tested. • Establish that the example problems represent actual problems for which the method is developed. • Establish the observations and data extracted from the example problems support the research hypotheses. • Consists of documenting that the example problems are similar to the problems for which the methods/constructs are generally accepted, that the example problems represent actual problems for which the method is intended, and that the data associated with the example problems can be used to support a conclusion.
Empirical performance validity	• Using the representative example problems to evaluate the outcome of the design method in terms of its usefulness. • The usefulness metrics is established based on requirements of the information model. • The solutions obtained and the outcome of the method is evaluated in relation to solutions obtained with competing / previous methods. • The data / solutions obtained through the proposed method and knowledge representation framework is demonstrated by critically evaluating the accuracy and internal consistency of the data. • The results obtained from the computer-based models (i.e., decision models and analysis models) are in agreement with aspects in the physical world.
Theoretical performance validity	• Established by showing that the method and knowledge representation is useful for design problems and applications beyond the example problems. • Generalize the characteristics of the example problems and show that the method and knowledge representation is useful for these problems.

The specific strategy developed for completing validation in this research is summarized in Table 1-7. Each of the validation quadrants are discussed in appropriate sections throughout the dissertation. In the following section, an outline of this dissertation is presented. The relevance of each chapter is established in the context of the validation strategy.

Table 1-7: Strategy for validation and verification for the information model

Theoretical Structural Validation
• Critical literature review and gap analysis of technologies and methodologies that are foundational to developing an information model. • The literature review is motivated by the primary research question and hypothesis posed in this dissertation. • The relevant topics reviewed in this research include multi-objective decision making and computational frameworks and information modeling formalisms and knowledge representations. • Identify and discuss the advantages, limitations, and accepted domains of application for available approaches? What are the opportunities for further work? In light of this critical review, do the research tasks and hypotheses represent original, significant contributions? • Gap analysis of current technologies for modeling design decision and integrating multiple design perspective. • Presentation and discussion of the information model for design decisions • Critically assess the advantages and limitations of the information model formalization.
Empirical Structural Validation
• Identify the significance of the example problems in the context of explicitly characterizing the knowledge associated with analysis models and design decisions taken. • Identify the need for a formal knowledge representation and method to support engineering decision making. • Discuss the appropriateness of the example problems in Chapter 4 (decision models), Chapters 5 (design of structural support members), Chapter 6 (structural heat sink). • Identify the characteristics of the examples problems in the context of the research gaps and research hypotheses.

Table 1-7: Strategy for validation and verification for the information model (continued)

Empirical Performance Validation
• Build confidence in the utility of the information model using the examples. • Does the information model address the research question in light of the example problem?
Theoretical Performance Validation
• Build confidence in the generality and utility of the approach beyond the specific example problems. Argue that the approach is useful for the example problems and that the example problems are representative of general problems.

1.6 Outline of dissertation

The flow and connectivity of the chapters, sections, and major themes are illustrated. The dissertation is organized according to the flowchart illustrated in Figure 1-11.

In Chapter 1, foundational research topics are established for the information model for decision-centric design. The motivation, frame of references, and overarching problem is established and presented. The overarching goal, research questions, and hypotheses are introduced and discussed in the context of the research problem. Finally, the expected contributions are summarized and the strategy for validating and verifying the method and knowledge representation is presented.

In Chapter 2, the theoretical foundations on which the information model and representation are developed are introduced including: engineering information management, multi-objective decision-making and decision-centric design, and product modeling. Theoretical structural validation is established by critically evaluating related literature. The current state of art and related literature is presented and the strengths,

limitations, and research gaps are discussed. These gaps serve as foundational motivation towards the development of the information model. The purpose of this chapter is to discuss in detail the strengths and limitations of the underlying methods, constructs, and technologies that are foundational to a knowledge-intensive approach to decision-centric design.

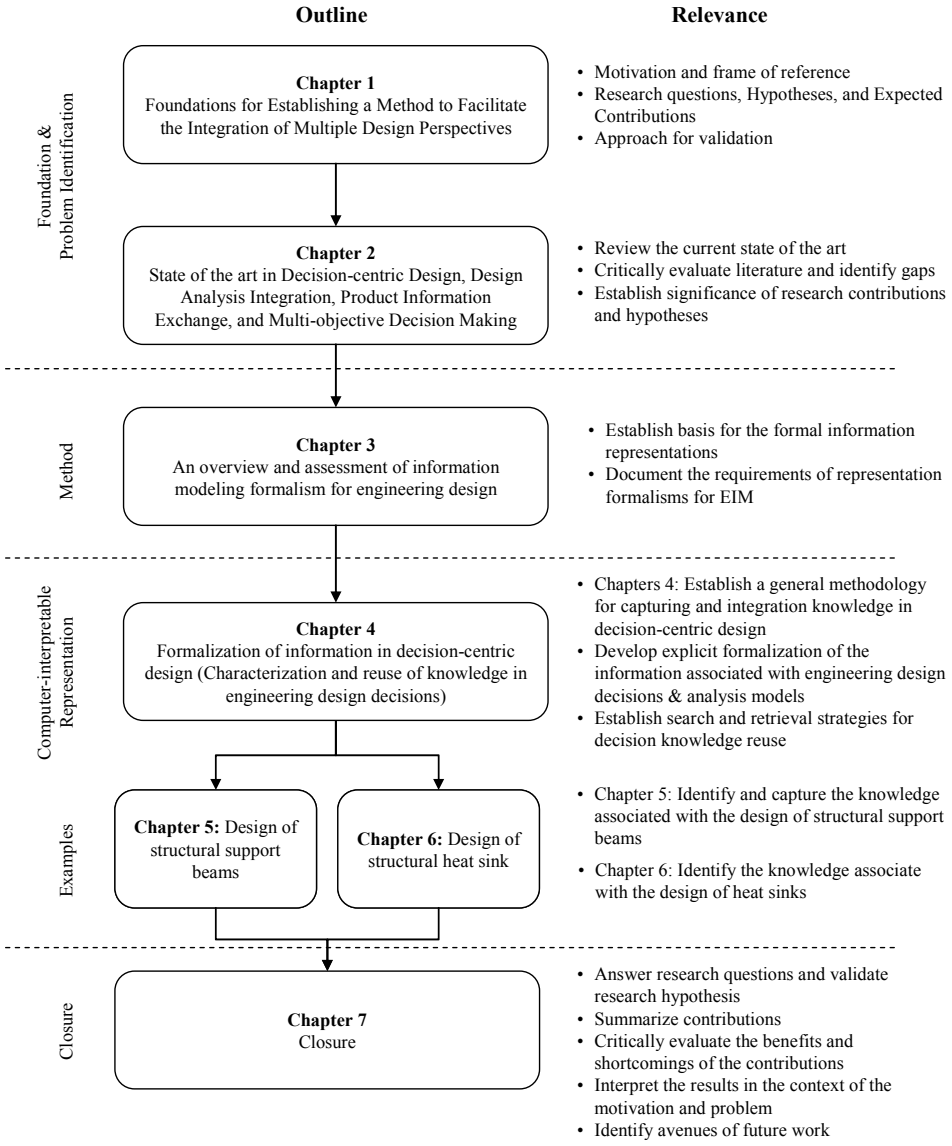


Figure 1-11: Outline of dissertation

In Chapter 3 information modeling formalism and computational-level building blocks are presented. An overview of information modeling is presented and discussed in the context of engineering design. Several information modeling formalisms are reviewed and assessed for modeling engineering design problems. An assessment of various formalisms is completed in the context of the characteristics and requirement of design problems. Techniques and technologies for developing knowledge representations in a computational infrastructure are reviewed and critically evaluated.

In Chapter 4, an overview of the information modeling framework is presented. A requirements list is created for defining the scope of the information model for capturing engineering design decisions. A systematic method is developed for formulating engineering design decisions based on the requirements and information modeling characteristics. An information model is developed for capturing decision making and analysis support models. The representation is then implemented in a computational environment using Description Logic. Advantages, limitations, and originality are discussed in relation to methods and constructs that are available in the literature.

In Chapters 5, and 6, two design example problems are presented. For each design example, problem statements are provided, and step-by-step implementation of appropriate aspects of the information model are discussed and documented. The results of the examples are presented, verified, and critically discussed for the purpose of empirical validation of the hypotheses introduced in Chapter 1.

In Chapter 5, the design of structural support members is presented to illustrate the need to capture and explicitly represent the knowledge associated with engineering design decisions. The decisions taken during the design of the structural support members

are developed and formulated using the compromise decision support problem as the underlying structure. Additionally, the knowledge associated with design decisions and the analysis models utilized to support the design decision are explicitly formalized and represented according to the ontology developed in Chapter 5. In Chapter 6, a structural heat sink design problem is presented to illustrate the use of the information model for multi-disciplinary design problems. Similar aspects are tested as in Chapter 5, but the level of complexity is increased.

In Chapter 7, the research completed in this dissertation is summarized and critically reviewed. Relevant contributions and future research opportunities are discussed. The advantages and application domains of the method and computational representation are discussed. Theoretical performance validity is presented and discussed in detail. It is asserted that the example problems used in this dissertation are representative of general problems and therefore the method and formal language is applicable beyond just the example problems.

CHAPTER 2:

CRITICAL LITERATURE REVIEW AND IDENTIFICATION OF RESEARCH OPPORTUNITIES

Aims

- To introduce and critically evaluate relevant literature
- To identify the gaps in current approaches
- To establish the research opportunities

In this chapter, the theoretical and foundational underpinnings are investigated for developing a model to capture the information associated with engineering design decisions. The research areas reviewed include decision-centric design, multi-objective decision making, decision making frameworks, product information modeling, design analysis integration, multi-aspect views of product models. The chapter concludes by identifying several research opportunities.

2.1 Representing Engineering Design as a Decision-Centric Processes

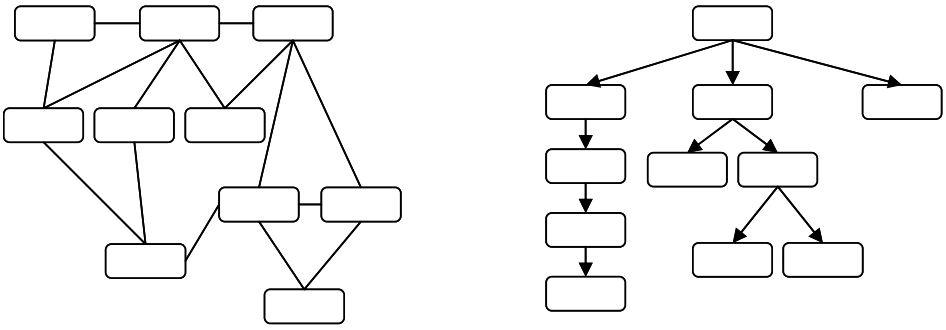
Panchal and co-authors [108] coin the term "decision-centric design" as an approach for representing design as set of design decisions and activities that support the decision-making process. Decision-centric design (DCD) is an overarching philosophy and mathematical approach for representing a design process in which the decisions encountered by engineering designers are formally modeled and serve as integrators between design disciplines and domains. A key characteristic of the mathematically-based decision constructs used to represent the product realization process include:

domain and discipline-independence. Thus, decision-centric design has advantages over other *centric* design perspectives in that decisions provide a common "language" to exchange design knowledge across multiple perspectives. Decision-centric design pervades all aspects of design and manufacturing [120]. For example, in the design of an aircraft decisions must be made about the final form and geometry based on strength, range, payload, speed, etc. Decisions are the "glue" that binds different aspects of the product realization process together and identified milestones in the design process.

Decision-centric design is based on Decision-Based Design (DBD) paradigm for modeling the product realization process. Mistree and co-authors state that DBD is a different perspective on which to develop methods to support design tools and techniques. Decision-Based Design, originally proposed by Mistree and co-authors [94], is a conceptualization of the product realization process in the context of domain-independent decision constructs. These domain-independent decision constructs provide a common platform, or "language", for modeling design problems that require the integration and composition of knowledge from multi-disciplines and multiple design perspectives [94]. Thus, stakeholders are able to collaborate throughout the product realization process at decision points. Decisions encountered in design can be described by the following characteristics [29]:

- Decisions involve information from multiple sources and disciplines
- All the information required to make a decision may not be available
- The information used in making a decision may be hard or soft information

The primary role of an engineering designer in DBD is to make decisions about the artifact that affect the outcome of the design process [93]. A core extension of DCD is the focus on design tasks and activities that generate information to support design decisions. In this context, designing is the process of converting information about the product at i into knowledge about the product at $i+1$ [90]. Designing a product becomes an issue of "information processing"; and a decision is an "irrevocable allocation of resources" that has two important characteristics: (1) a decision is made at an instant in time and (2) a decision must be made with the information available at the time it is made. Bras [29] states that *"designing is a process of converting information that characterizes the needs and requirements for a product into knowledge about a product."* In designing this artifact many decisions may be made, either in parallel or series, which when viewed collectively represent the design process. Decisions may be hierarchically organized from the component to the sub-systems to the system-level (see Figure 2-1).



Hierarchical Decision Representation

Heterarchical Decision Representation

Figure 2-1: Heterarchical and hierarchical representations [94]

Design decisions are may rely on varying levels of quantitative or qualitative information, depending on where in the design timeline the decisions are made. For example, at the conceptual stages of design, decision may be based primarily on

qualitative information, whereas at the end of design decisions may be based on quantitative information. Design processes can be modeled through decisions that serve as markers for progression from design initiation to completions and as a unit of communication and support tasks that create, manipulate, and modify information to support design decisions.

The principle point abstracted from DCD is that the engineering product realization process is represented as a set of inter-related design decision and supporting activities which serves as units of communication and collaboration between stakeholders in the design process. Decision-centric design facilitates the integration of multiple design perspectives by establishing a common construct through which designers from various perspectives can integrate and reconcile knowledge related to a particular design problem. As previously stated, the challenges that arise in multi-disciplinary, multi-perspective design are addressed in this dissertation through a decision-centric design (DCD) approach. From a conceptual level, the DCD paradigm provides a unified view of the product realization process that enables information to be seamlessly shared across design discipline. However, the question arises of how to represent the decision commonly encountered in design? The implementation taken in this research is the Decision Support Problem Technique (DSP Technique) for representing engineering decisions and design processes.

2.2 The Decision Support Problem Technique

The Decision Support Problem (DSP) Technique [28; 90; 93-95; 99] is proposed as a specific implementation of DCD. The DSP Technique is rooted in systems thinking, and

approach for formulating DSPs in engineering design. The DSP Technique compromises several components, including:

- A method for representing design processes – the DSPT Palette
- A method for formulating decision from a lexical and mathematical formulation
- A computing environment for integrating tools

As discussed in the previous section, engineering design decisions are made based on information from many different disciplines. While there are computer-based tools to aid in decision-making, these tools become increasingly inefficient and ineffective as the complexity of the problem increases and the perspectives in which these products are viewed. In addition to these troubles that arise, the relationships between the various disciplines involved make it impossible to take into account the interactions in a rational way. Muster and Mistree state the key concepts of decision making as the following:

- Design is a decision-making process in which it is preferable that some of the decisions be made sequentially and others be made concurrently
- Design involves hierarchical decision-making and the interactions between decisions, if any must be taken into account
- Design productivity can be increased by the use of analysis, visualization and synthesis in complementary roles
- Design productivity can be increased through use of analysis and synthesis and their complementary roles

The DSP Technique is proposed as an environment in which the harmony, balance, and interactive co-operation we seek among designers, their approaches and their computers are an integral element in the design process [99]. The DSP Technique provides support for human judgment in the decision making process. In other words, the DSP Technique provides a means for modeling decisions that leverages the abilities and knowledge of the human designers and the capability of computer-based models and applications. The DSP Technique enables the efficiency and effectiveness of the design process to be increased by reducing the number of decision iterations, increasing the speed of the iterations and making tools available for modeling the product realization processes in a computer-based environment.

The DSP Technique consists of two phases: meta-design and design. The first phase is a "meta-design" phase where the necessary decisions in the design process of a product are identified and structured with based on several axioms for modeling engineering design decisions [95]. In meta-design, the problem is partitioned and design problems are identified in terms of decision problems. The design process may be partitioned into a set of design problems that are modeled as design decisions, and the sequence in which the decisions are carried out is established. In the meta-design phase, specific decisions are not completed, but rather the decisions are structured. Information sources needed to complete the design decisions are modeled and links between design decisions are established. In the second phase, the domain dependent information is formulated and structured in accordance with established decision constructs. In the actual design phase, the design process is “executed” and solutions are determined. However, an important area that has received little attention is the development of design environments to

integrate available tools. As alluded to previously, computational tools and standardized interfaces are needed to support the decision making process.

A key component of the DSP Technique is the DSP Technique palette (DSPT Palette). The DSPT palette provides entities for creating graphical networks for modeling design processes with a focus on engineering design decisions. The entities can be used to model design process in a domain-independent manner. The DSPT Palette contains three different classes of entities – potential support problem entities, base entities and transmission entities (see Figure 2-2). The DSPT palette entities can be used to model a design process at varying levels. A detailed discussion and examples are provided in [94].

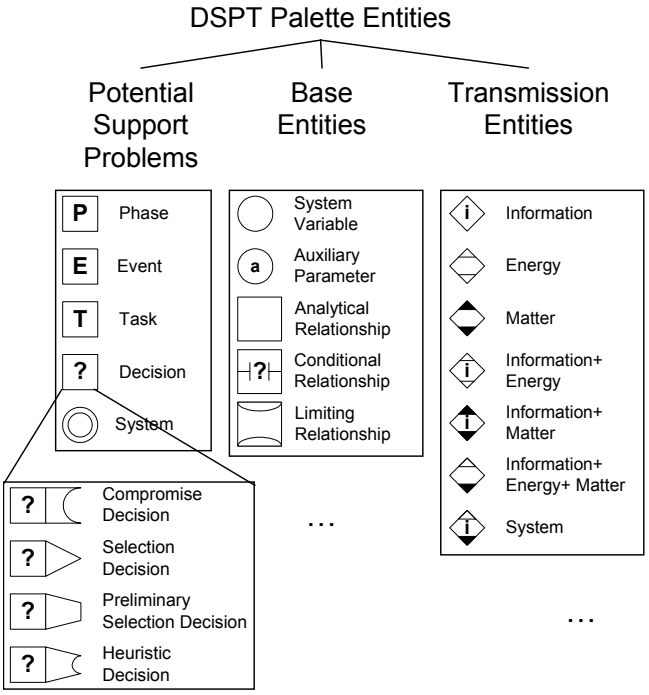


Figure 2-2: Decision Support Problem Technique Palette

As previously stated, Decision Support Problems (DSP) form the foundation of the DSP Technique [18; 94]. DSPs are mathematical constructs for modeling engineering design decisions. DSPs provide the structure and organization of relevant decision-

making information to model decisions commonly encountered in a product realization process. The formulation and solution to DSPs enable designers to model several types of decisions typically encountered in design. Mistree and co-authors [94] assert that there are two primary types of decisions: the selection decision support problem [18; 49; 92] and the compromise decision support problem [91]. Complex design decisions, such as coupled decisions [18], can be derived from these basic decision problems. A selection decision involves the choice between a discrete number of options and a compromise decisions involves adjusting the values of design variables to result in the "best" feasible results for a certain design objective. In the DSP Technique, the selection decision is the process of making a choice between a number of possibilities taking into account a number of measures of merits or attributes. In selection, designers reduce the number of alternatives to a realistic and manageable number based on different measures of design attributes of interest. The compromise decision requires that the 'right' values (or combination) of design variables be determined, such that, the system is feasible with respect to constraints and the deviation from the target design objectives are minimized. The emphasis on compromise is on modification and change by making appropriate tradeoffs. The goal of compromise in design is that of modification through iteration based on criteria relevant to the feasibility and performance of the system. Decision support problems are modeled in a computational environment by establishing keywords and descriptors to describe the information and knowledge within the decisions. The keywords and descriptors for the cDSP are summarized in Table 2-1.

Table 2-1: Keywords and descriptors for Compromise Decision Support Problem

Keywords	Descriptors
Given	Symbolic and mathematical base entities and Support Problems necessary for evaluating the goals, constraints and bounds and the deviation function
Find	System variables (symbolic and mathematical)
Satisfy	Goals, constraints and bounds, i.e., symbolic and mathematical relationships
Minimize	A Deviation function

The DSP Technique provides a conceptual basis on which to model product realization as a decision-centric process. However, in a large part the DSP Technique exists as a conceptual, graphical tool for representing the design process. The decision support problems are the only constructs that have been adequately and formally represented in a computational environment at the mathematical level. However, the semantics of the design decisions and the integration and representation of knowledge from multiple perspectives has not been fully developed.

2.3 The Compromise Decision Support Problem for Modeling Engineering Design Decisions

The compromise decision support problem (cDSP) is a hybrid formulation of mathematical and goal programming. In the most general form, the conventional mathematical programming problem is formulated as follows:

$$\begin{aligned} &\text{Minimize} && f(\mathbf{x}) \\ &\text{Subject to} && g(\mathbf{x}) \le 0 \\ & && h(\mathbf{x}) = 0 \\ & && \mathbf{x}_{lb} \le \mathbf{x} \le \mathbf{x}_{ub} \end{aligned}$$

where $f(\mathbf{x})$ is the objective function to be minimized by varying the set of design variables $\mathbf{x}$, $g(\mathbf{x})$ and $h(\mathbf{x})$ are vectors of inequality and equality constraints, respectively and $\mathbf{x}_{lb}$ and $\mathbf{x}_{ub}$ are the lower and upper bounds on the vector of design variables. When considering multiple objectives, the objective function is represented as a vector and expressed as:

$$\text{Minimize} \qquad f = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\} \qquad \textbf{2.1}$$

where m is the total number of objectives. By placing different priorities on the individual objectives, it is possible to obtain many different solutions to the multi-objective problem. The range of compromise solutions is often called a Pareto set, curve, or frontier. Solutions along the Pareto curve are defined as non-dominated, meaning there is no other solution in the feasible space that improves one or more objectives without worsening the others.

Design solutions are rarely judged on the basis of a single objective, rather on how well they balance multiple criteria associated with cost, performance, environmental impact, robustness, and other categories. Therefore, in order to balance between multiple objectives many techniques have been proposed for generating the Pareto set. The simplest and most straightforward technique is a linear weighted sum approach, where the weighted sum of the objective function is expressed as a linear additive combination of the multiple objectives.

$$Z = \sum_{i=1}^{m} w_i f_i \qquad \textbf{2.2}$$

where w_i is the weight for the i^{th} objective, f_i, and m is the number of objectives. This approach is simple to implement and understand. The weights on the objectives can be varied, thus it is possible to generate a family of Pareto solutions. However, foundational criticisms include overlooking some Pareto solutions and difficulties in determining *a priori* an appropriate set of weighting coefficients if a single objective is sought.

In order to address these problems, the compromise decision support problem is proposed as a mathematical construct for modeling multiple objectives in engineering design applications based on goal programming and mathematical programming. The focus of goal programming is to establish goals for each objective and to achieve each of the goals as closely as possible. For each objective, an achievement function, $A_i(\mathbf{x})$, represents the value of the objective function as a set of design variables, $\mathbf{x}$, and the goal value, G_i, for each objective. Deviation variables, d_i^+ and d_i^-, represent the extent to which an objective is either underachieved or overachieved. The goal function is represented as follows:

$$A_i(\mathbf{x}) + d_i^- - d_i^+ = G_i \qquad \textbf{2.3}$$

The overall objective function is expressed as a function of the deviation variables as follows:

$$Z = \underset{i=1,\ldots,m}{f}\left(d_i^-, d_i^+\right) \qquad \textbf{2.4}$$

The basis of the cDSP is to minimize the difference between what is desired, G_i, and what is achieved, $A_i(\mathbf{x})$. This difference is calculated based on the total deviation represented by the deviation variables, d_i^+ and d_i^-. Leveraging from the weighted sum

approach, presented in Equation 2.2, the objective function is used to aggregate multiple objectives into an Archimedean objective function based on deviation values:

$$Z = \sum_{i=1}^{m} \left(w_i^+ d_i^+ + w_i^- d_i^- \right) \quad \textbf{2.5}$$

It is impossible to simultaneously overachieve and underachieve a goal, thus restrictions are placed on the deviation variables to limit them to positive values and ensure that only one deviation variable is positively valued at any specific point in the design space:

$$d_i^- \geq 0;\ d_i^+ \geq 0;\ d_i^- \bullet d_i^+ = 0 \quad \textbf{2.6}$$

Although strict formulations of goal programming do not support equality or inequality constraints, these constraints are supported in the cDSP with formulations borrowed from mathematical programming:

$$g_i(\mathbf{x}) \geq 0,\ i = 1, ..., p \quad \textbf{2.7}$$

$$h_i(\mathbf{x}) = 0,\ i = 1, ..., q \quad \textbf{2.8}$$

where p and q are the numbers of inequality and equality constraints, respectively. Bounds are also specified on the set of control variables that describe the form of potential solutions:

$$x_{i,lb} \leq x_i \leq x_{i,ub},\ i = 1, ..., n \quad \textbf{2.9}$$

where n is the number of design variables and $x_{i,lb}$ and $x_{i,ub}$ are the lower and upper bounds, respectively, for the i^{th} variable. The objective function formulation and constraints borrowed from goal programming and mathematical programming,

respectively, are unified into a single mathematical construct for decision support, the cDSP template. The mathematical formulation of the cDSP is illustrated in Figure 2-3.

Given

n	number of system variables	
$p+q+r$	number of system constraints	
p	linear constraint	
q	nonlinear equality constraints	
r	nonlinear inequality constraints	
m	number of system goals	
$g_i(X)$	system constraint function $g_i(X) = C_i(X) - D_i(X)$	
$Z_k(d_i)$	function of deviation variables	

Find

X_i	System variables,	i = 1, …,n
d_i^+, d_i^-	Deviation Variables	i = 1, …, 2m

Satisfy

System Constraints

Linear constraints $g_i(X)\ (=\text{or}\leq\text{or}\geq)\ R_i$	i = 1, …,p
Nonlinear equality constraints $g_i(X) = 0$	i = 1, …,q
Nonlinear inequality constraint $g_i(X) \geq 0$	i = 1, …, r

System Goals (linear, nonlinear)

$A_i(X) + d_i^- - d_i^+ = G_i$	i = 1, …, m

Bounds

$X_{i,L} \leq X_i \leq X_{i,U}$	i = 1, …, n
$d_i^+ \cdot d_i^- = 0$	i = 1, …, 2m
$d_i^+, d_i^- \geq 0$	i = 1, …, 2m

Minimize

Deviation Function: Archimedean formulation

$Z = \sum_i W_i\left(d_i^+, d_i^-\right)$ i = 1, …, m

OR

Deviation Function: Preemptive formulation

$Z = \left\{Z_1(d_i^-, d_i^+), \ldots, Z_k(d_i^-, d_i^+)\right\}$ i = 1, …, m

Figure 2-3: Mathematical formulation of cDSP [91]

The cDSP is used to determine the values of design variables that satisfy a set of constraints and bounds and achieve a set of conflicting, multifunctional goals as closely as possible. The conceptual basis of the cDSP is to minimize the difference between that which is desired (the goal, G_i) and that which can be achieved ($A_i(x)$) for *multiple* goals.

The underlying philosophy of the cDSP and its goal programming foundations is similar to the concept of *satisficing* solutions and bounded rationality proposed by Simon [140].

The cDSP has been previously developed and demonstrated in a variety of domains, thus the contribution in this dissertation is on the development of a computational infrastructure to support the formulation of compromise decision support problems from multiple design perspectives. As previously discussed, the cDSP is a mathematically rigorous construct for modeling multi-objective design decisions. The cDSP has been applied and augmented for problems in several domains. For example, the cDSP has been utilized in specific design applications such as the design of ships [94], rapid prototyping manufacturing and parts[38; 131], and micro-electronic-mechanical (MEMs) devices [132]. Additionally, the cDSP has been extended with the infusion utility theory [49; 133], uncertainty [39], game theory [168], and coupled decision making, [70]. Additionally, there is continued research in the areas of material design, robust design, and resolving conflicts between designers and design space reduction to name a few. However, a fundamental shortcoming in the cDSP in particular and in multi-disciplinary optimization in general is the underlying computational infrastructure to support the integration of knowledge from multiple disciplines. Related efforts are summarized in Table 2-2.

Table 2-2: Summary and comparison of cDSP augmentations and developments

Baseline cDSP [91; 95; 99]
• Convenient, discipline-independent means for exchanging decision-making information • Enables designers to model decision to determine the "right" values (or combination) of design variables (e.g., system parameters) • Provides a common "language" for representing decisions in the design process • Does not support the integration and reuse of design knowledge encoded in computer-based models • Does not capture the complex mapping between design and analysis spaces • Requires all relevant decision-making information to be encoded within the cDSP template • Provides broad guidelines (word formulation) for developing decision templates – specific computer-based construct do not exist
Information Modeling of Engineering Decisions [25]
• Provides foundation for modeling and archiving decisions in a "digital" representation • Supports the collaborative development of decisions • Does not leverage from current design knowledge representation • Does not directly support the integration of knowledge form external sources such as computer-based product models
Geometric and Materials and Process Tailoring [38; 126; 130; 131]
• Enables computer-based models to be loosely "packaged" with decision information • Facilities geometric manipulation through decision constructs • Provides a means for integrating various design domains – including design and manufacturing • Behavioral models and analysis are limited to relationships that can be represented within the decision
Collaborative Decision Making in a Distributed Environment [70; 168]
• Computer-based models are "linked" with design information • Support collaborative decision making through the infusion of game theory • Provides a means for integrating various design domains • Does not provide access the knowledge within models, thus true integration between design perspectives is limited
Decision-based Design Templates for Modeling Design processes [108]
• Addresses the reuse of decision at the enterprise level • Enable modular creation and reuse of decision • Towards the development and implementation of the DSPT palette • Does not address the micro-level relationships between different design perspectives • Does not provide patterns for specific decision templates for various design perspectives

Karandikar and co-authors [71] develop an object-oriented data model as a means for capturing decision-related information, exchanging design information, and document control. Bollam [25] extends the conceptual work of Karandikar through the development and implementation of a database schema for storing decision support problems. The database and corresponding information model enables designers to create decisions by cooperatively populating a template that is stored in a centralized database. Despite advances in knowledge representations (i.e., ontologies [56; 60; 83; 115; 139], information models for engineering [97; 98; 122; 131; 145; 147; 148], and design repositories [38]) there has been little towards extending Bollam's work for capturing the semantics and integrating knowledge from multiple design perspectives.

Sambu [130; 131] and Rosen [126] address problems with rapid prototyping (RP) and rapid tooling (RT) in the context of design for manufacturing (DFM) frameworks. A "global cDSP" template that combines coupled and domain specific cDSPs from different domains in RP and RT is proposed, that is then decomposed into decoupled cDSPs and subsequently solved. Chen [38] further builds on cDSP templates for rapid manufacturing by developing decision templates for geometric and process tailoring. The Material Geometric Tailoring (MGT) template and the Material-Process Geometric Tailoring Decision Template (MPGTDT) are developed to facilitate collaboration between designers and manufacturers by capturing design requirements of functional prototypes for geometric modeling and material and process selection. Rosen, Sambu, and Chen assert that the proposed decision templates support the integration of CAD and FEA models along with information, such as constraints, bounds, and goals, for making decisions. However, the decision templates support the same level of *model integration*

between multiple design perspectives. The cDSP templates simply serve as "wrappers" for packaging computer-based models, such as CAD and FEA models, with decision-making information. The models, and thus the various design perspectives, are loosely integrated with decision making information, requiring designers to manually extract and recreate product information and knowledge in the cDSP. Xiao [166; 168] proposes the Collaborative Multidisciplinary Decision-making Method (CMDM) to bridge the gaps in multi-disciplinary product realization. The relevant decision-making information is "packaged" in cDSP templates to facilitate information exchange and collaboration between design perspectives. The cDSP is a convenient, discipline-independent means for exchanging decision-making information between design activities, but product information, such as complex geometric information cannot be represented. Thus, computer-based product information, such as data files or databases, can be "linked" to or "embedded" within the decision templates.

Additionally, in a recent research effort by Panchal and co-authors [109] decision templates are proposed as a means to reuse decision-related information. Panchal [107] and Panchal and co-authors [108] propose the development of reusable, executable building blocks for modeling decision-based design. This research is motivated by the extension and refinement of the DSP Technique. In their research, the reusability and scalability of decision-based design process is explored. Decision templates are created by developed XML schemas that capture decision-related information. However, the authors do not leverage from the early conceptual modeling techniques, nor do they address the notion of capturing semantics from multiple design perspectives. The reusable decision templates for modeling design process from a decision-centric

perspective enable designers to: (1) specify design processes from reusable modular building blocks, (2) execute design processes based on computer-interpretable building blocks, (3) archive the design processes for reuse, and (4) analyze design processes. Specifically, a modular architecture is proposed for formulating and reusing design decisions. The proposed architecture is analogous to printed wiring boards (PWB).

While the cDSP has been applied and augmented in a variety of ways, a mechanism to support the *information-intensive* integration and mapping of computer-based design and analysis models between design perspectives. The cDSP has been extended to support the "loose" integration of computer-based product models from multiple perspectives at the model-level, but does not support the detailed associativities between models and decisions. The strengths and limitations of the cDSP are presented in (see Table 2-3).

Table 2-3: Summary of strengths and limitations of baseline and augmented cDSP

Strengths
• A construct that provides high-level guidelines for modeling & implementing design decisions in DCD • Provides decision support for achieving compromise among multiple goals, constraints, and bounds • Facilitates modeling design along a timeline • Enables designers to model decision to determine the "right" values of system design variables • Convenient, discipline-independent means for exchanging decision-making information
Limitations
• Does not support the integration of computer-based simulation and design models • Does not support the representation of mapping and associativities between design perspectives • All relevant decision-making information must be encoded within the cDSP template • No "standardized language" to facilitate communication between across decision constructs • Does not provide computational-level specifications for formulating decisions

The cDSP construct provides designers with a means for representing design decisions through an established structure. Similar to IDEF0, the cDSP enables designers to model the design process from a conceptual view in the context of engineering decisions. However, the cDSP does not support the need to *capture and represent the information intensive nature of engineering design perspectives and the associated computer-based product models*. For example, an obvious shortcoming is the inability to "link" external models to the cDSP construct such as CAD, FEA, or CFD models. In part this is an implementation issues. However, there are fundamental questions that arise when considering the integration of knowledge from external models relating to the communication and exchange protocol between decision construct and associated support models. To address this problem, a language must be established with agreement on how to communicate, syntax, and what to communicate, semantics. The limitations of the cDSP, as summarized in Table 2-3, must be addressed to realize the integration of multiple design perspectives. Maturing and emerging technologies in information and knowledge modeling offer the ability to create such a language that can be shared by multiple stakeholders in the decision-making process. The implicit assumption in modeling multiple objectives and integrating knowledge from several perspectives is that a single designer will coordinate and be knowledgeable in all relevant areas. However, this is not always the case, as distributed experts may be responsible for a particular aspect or characteristic of the product. The cDSP does not support the need to capture and represent the information-intensive nature of engineering design perspectives and the associated computer-based product models. A critical review of applications and

augmentations to the baseline cDSP and a gap analysis is completed to identify the research opportunities in the following section.

2.4 Product Information Exchange in Engineering Design

The need for engineering information management (EIM) systems and technologies is motivated by: (1) the immense amount of digital information generated in design and (2) the need to share the information across the extended enterprise. As previously stated, information exchange in the design of complex systems is difficult and hindered by interoperability problems. For example, interoperability and information exchange problems in the automotive supply chain are estimated to cost nearly $1 billion U.S. dollars per year [31].

EIM technologies have both helped and hindered information capture and exchange. Existing software tools do not address information exchange and coordination, but rather increase problems by isolating product information at the boundaries of the specific tools, resulting in a *Tower of Babel* [61]. For example, it is difficult, if not impossible, to directly exchange geometric information between CAD systems [128].

Product information modeling is an essential step in the development of databases and information management systems. A product model is a specialized data model that captures information that is relevant to the product life cycle. Data information modeling is a process of specifying the structure of information in a particular domain. In data modeling, the goal is to develop a representation that explicitly captures *things*, attributes of those *things*, and relationships between *things* in the domain of engineering design. Data modeling has received significant attention in the areas of databases, information

systems and knowledge representations. Business-process database design and development has provided foundations for data modeling that can be applied to engineering design problem to a limited degree. EIM is relatively new in the field of engineering design because of the complexity of information and a different set of requirements and characteristics over business-process applications [53]. Information modeling provides an explicit specification of the semantics in a domain, thus reducing the communication problems between designers. Information modeling is important in addressing the second requirement for the computational framework. The second requirement is that the framework should provide a computer-interpretable means for representing knowledge that can be exchanged and shared amongst stakeholders in the decision making process.

Modern engineering design is largely a digitally-based activity, one in which product information is created, manipulated, viewed, and modified at various levels of abstraction. Design is a collaborative set of tasks among multidisciplinary, distributed design teams in which communication and collaboration is facilitated by extended network and computer-based design support tools [149]. The use of computer-based design support tools, extended networks, and engineering information management technologies is increasing. In the context of distributed, collaborative decision-making, computer models are used for simulating product behavior and representing product geometry. In the design and development of large scale, complex products designers often decompose the product into different "views" and focus on a subset of product information that corresponds to their domain and design concerns [46; 96; 127; 128]. Decomposing a system into smaller systems is often required to address complex issues,

but can lead to information integration problems. Information integration and interoperability problems can be attributed to inadequate and insufficient computer-based representations of product information. This is especially true in computer-based design and engineering information management [80]. For example, interoperability and information exchange problems in the automotive supply chain are analyzed and estimated to cost nearly $1 billion U.S. dollars per year [31]. The problems associated with information exchange and interoperability in engineering design are becoming well-understood. Information must be viewed as a key integrator in product development to alleviate many of these problems [53]. A number of research and development efforts in industry, government, and academia are actively addressing the problem. Specifically, researchers have pursued the development of computer-based product and process models for exchanging information throughout the development process.

Thus, engineering information management (EIM) is becoming increasingly important for developing technologies and representations to enable the exchange of product information. In this context, EIM broadly refers to the methodologies, software support tools, and product information models that enable engineering design information to be operated upon throughout the design process. The need for EIM systems and technologies is motivated by: (1) the immense amount of digital information generated in design and (2) the need to share the information throughout the design chain. Fulton and Chadha [53] assert that EIM systems are essential to manage and integrate product data generated throughout the product realization process. EIM systems and technologies have received significant attention to alleviate the *interoperability* and information exchange problem in engineering design, thus increasing the efficiency and effectiveness of

collaboration in design. In this context, efficiency is a measure of the speed at which information is accessed and used and effectiveness is a measure of the correctness of the information for a design activity.

Computer-based engineering support tools are essential in EIM, but are a double-edged sword that have both helped and hindered engineering design. On one hand, computer-aided design, manufacturing, and engineering (CAD/CAM/CAE) software tools have enabled designers to capture product information digitally for particular design activities. CAD/CAM/CAE tools are used for geometric modeling, structural and thermal analysis, and manufacturing planning. On the other hand, the tools and underlying data structure are often developed independently and focused on a particular problem, resulting in a *Tower of Babel.* For example, the use of CAD in product design is becoming ubiquitous for creating digital product representations that capture product form [128]. However, it is difficult, if not impossible, to directly exchange and share geometric information between CAD systems. Additionally, sharing CAD information across distributed networks is not adequate to fully support the development of modern engineering systems, as a CAD model can only provide a small subset of the total product-related information [147].

Standardized information models and data sharing languages, such as the eXtensible Markup Language (XML), have decreased interoperability gaps in software tools by enabling data to be shared amongst heterogeneous software applications and distributed agents. Standardized information models usually address the syntactic aspect of interoperability. While standards usually address a small portion of the total interoperability problem syntactic translation represent a significant difficulty in design.

Thus, the motivating problem that standards address is *how product data can be shared between disparate heterogeneous software applications in a transparent and efficient manner*. In particular, ISO 10303 (commonly known as STEP) has provided a set of standardized product models for exchanging engineering product data between software applications used throughout the product life-cycle [4; 57; 65]. The overarching technical objective of STEP is to enable the communication and exchange of product data between heterogeneous design software systems. STEP is - "a neutral mechanism capable of completely representing product data throughout the life-cycle of a product,...The completeness of this representation makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing databases and archiving." [32]. Standards-based product models provide a mechanism to enable the exchange of data between heterogeneous software applications [4; 57; 65]. The development of the STEP was initiated in 1984 and is one of the largest efforts undertaken by ISO. It was initially born out of the need to share product geometry between CAD systems, but the scope has quickly expanded to include many different design disciplines. STEP is increasingly becoming recognized as an effective means of exchanging product-related data between different CAD systems or other life-cycle support systems.

STEP aims to eliminate many of the problems associated with integration, by providing a method to exchange and share rich product information. The standard allows for platform-independent sharing and exchange of product data information [4]. Additionally, adhering to a standards-based approach for exchanging product data reduces the effort and cost to exchange data, reduces human error by providing software support for data exchange, and provides product data to all the stakeholders in the design

process. For example, Kemmerer and co-authors [4] cite the economic and technical impact of standards-based product models resulted in a 27% saving on CAD/CAM systems for Lockheed Martin and 75% time saving in the development of the Boeing 767 and 777 programs across the extended design-manufacturing enterprise. The economic study of standards-base product models, completed in 2002, estimates the impact in aerospace, automotive, and shipbuilding to be approximately $928 million dollars annually [54]. However, there are several limitations of STEP. First, STEP is focused on describing and exchanging finished designs rather than on supporting designs in progress [162]. STEP does not provide the facilities to track the changes in engineering information and the evolution of the product. STEP models are a snapshot of the product in time and do not capture how the product was changed or the decision taken to determine the product specifications. The scope and information content of the information models are difficult to determine and establish as a standard because engineering information is dynamic, complex, and diverse. The common argument against STEP is the length of time for the development and implementation of standardized product models in commercially available design tools. This argument was increased when XML was first popularized as a means for exchanging data. However, comparing XML to STEP is incorrect in the sense that XML provides a means for exchanging data and STEP is provides the schema for modeling the information structure in a very complex domain, i.e., engineering design and manufacture. STEP enables product design information to be shared between design support tools by addressing syntactic translation problems, but does not provide support for capturing complex relationships between design disciplines.

To address the shortcomings of STEP, a number of information models have been proposed. Several product models have been proposed to capture information over the life-cycle of the product. For example, the core product model (CPM) [43; 44], the open assembly model (OAM) [84], the design-analysis integration model (DAIM) [46] and the Product Family Evolution Model (PFEM) [45] are proposed as a means to capture additional product information beyond the STEP standards. The collection of models is the central information models for the PLM framework to enable interoperability and exchange of product information in design support systems and tools.

Design-manufacturing enterprises need to exchange, collect, and organize product-related design knowledge to facilitate efficient and effective reuse. Information modeling provides the basis for developing EIM systems to support information reuse. Reuse in engineering design ranges from reuse of design process activities to reuse of physical parts and standardized components. There are several classifications of design reuse which are useful for exploring the reuse and exchange of design knowledge. First, Pahl and Beitz [105] classify design into three categories of design, based on functional relationships in the system.

- Original design – the functions and/or sub-functions of the system and therefore the structure, assembly, and components are not known. Typically designers start with a requirements through which functional relationships are abstracted to determine the function structure.
- Adaptive design – the general structure, assembly, and components are better known, leading to the determination of unction structure. The function structure for the

system can be modified by varying, adding, or omitting individual functions. The starting point is the function structure of an existing design solution.

- Variant design – uses well established modular components. The assemblies and individual components are well known and are reflected in the function structure of the system. The function structure is examined for determining modular design.

Design repositories are a knowledge-based approach for supporting engineering design by enabling design manufacturing enterprises to capture and store design knowledge. Design repositories are intended to store information in a reusable and rich way. They are not intended to be catalogs of parts, but they should provide more knowledge than just that. For example, design repositories not only capture *what* is designed, but also *how* and *why* the product is designed. Design repositories enable engineers to capture the evolutionary nature of product knowledge and information throughout the design process [147]. Kopena [73] explores the use of design repositories for capturing domain knowledge. It is argued that it is essential to develop mechanisms to capture knowledge from disparate sources to support engineering design to address a number of challenges that include:

- Integration of design knowledge from disparate sources - repositories must facilitate data collection from multiple sources. Input from multiple sources must be integrated in order to develop a cohesive collection of product-related design knowledge.

- Assess and management of design knowledge – design repositories must support access and utilization by several mechanisms. Querying and retrieving knowledge must go beyond simple keywords to a higher level of sophistication.

- Scalability of the design knowledge – the structure and organization of design knowledge and information cannot be determined a priori. A primary challenge is to provide sophisticated reasoning and querying services to support large, diverse knowledge bases.

There are several examples of design repository research to support product development. Grosse [58] presents an ontological design repository for capturing knowledge about behavioral models in engineering. Similarly, research in [85; 97] propose the development of design repositories for formally characterizing analysis models in design to support analysis reuse beyond "black box". The National Design Repository Project [121] is a web-based repository for capturing engineering knowledge. The repository provides facilities to capture similar attributes with the addition of file types and keywords. The design repository developed at the University of Missouri-Rolla [21-24; 143] provides designers the capability to capture design knowledge and search and browse the repository by attributes including color, physical parameters, manufacturing process, part number, and assembly information. In addition, a primary focus of the repository is on capturing and representing function-based information to enable designers to reuse design information based on desired system and sub-system functionality. The repository is closely related to the design process developed by Pahl and Beitz [105] and thus adheres to elements such as function structures and morphological design matrices. Designers are able to select design components and systems based on a number of functions and input and output flow. Kopena and Regli [74] propose a similar design repository for capturing function-based information through the development of description logic-based knowledge representations.

Engineering databases and design repositories have been discussed as independent systems. For example, Szykman lists several distinctions between engineering databases and engineering design repositories [149]. Unfortunately, the distinction between engineering databases and design repositories has gained unfounded momentum. A common claim made against engineering database research is the limited expressiveness and retrieval capabilities of design knowledge and providing the facilities for a single type of data. In this research, the authors believe that the distinction between engineering repositories and databases is a secondary issue. The primary objective in engineering design repository research should be identifying what design knowledge should be captured how it is used to support computer-based product realization. For example, Kopena [74] asserts that "search is limited to keyword scanning or matching on overly simplistic attributes." As previously discussed commercially available PDM software provide the facilities to store design documents based on simple attributes. However, this limitation is not a result of engineering databases, but rather the scope of the conceptual information model of the domain.

2.5 Computational Frameworks for Engineering Design Decisions

As previously stated, a fundamental challenge in design decision making is facilitating collaboration and information exchange between multiple design perspectives and disciplines. Advanced computational technologies are needed to support engineering decision making to enable the integration of information from multiple sources [17; 55; 129; 141]. Several frameworks and languages have been developed to enable multi-objective decision making to varying levels of success.

For example, the Framework for Interdisciplinary Design Optimization (FIDO) system, developed at NASA, is a computer-based framework to support heterogeneous distributing computing for design optimization by "linking" heterogeneous models to support design optimization. FIDO is a modular architecture using “wrapper” technology for legacy analysis codes [154]. The Virtual Airplane Design Optimization Framework (VADOR) [34; 103] is based on object-oriented technology using the Java programming language to ensure the framework is portable and internet capable. Peer-to-peer technology is used to enable individual designers to communicate. User interfaces are specified to enable data flow between components in the framework. Design and analysis data are encapsulated in objects and described by a set of attributes to enable data management. The use of the VADOR framework is illustrated using computational fluid dynamics (CFD). Vander Kam and co-authors [157] develop a semantic linking agent based a commercially available framework that automatically creates “linkages” between agents during the design and analysis process to support multi-objective decision making. The language enables relationships between common elements to be created.

Web-DPR is a web-based software framework to support the integration of heterogeneous software applications in design and manufacturing [167]. Web-DPR is a conceptual framework for distributed collaborative product realization. Software agents and information can be exchanged between engineers in Web-DPR by wrapping data in XML-based templates and provide access to multiple sources of information to support the design process. The rapid tooling testbed (RTTB) [126] is implemented on the Web-DPR system. To deploy the RTTB several decision templates are developed [125]. The decision templates specify the structure and overarching “schema” of information

required to complete a design decision, in this context a design-for-manufacture decision. X-DPR is based on the Web-DPR framework. The X-DPR framework is a computer system that allows designers to capture and complete meta-design of distributed product realization processes in accordance with the DSP Techniques, discussed in Section 2.2. The flow of information between agents is encoded in a standard XML interface and exchanged via the SOAP protocol. X-DPR is an open system in which different modules can easily be integrated to the system for enhancing the overall functionality. The X-DPR framework enables a variety of applications and models to be "wrapped" and linked together to support the engineering decision making process (see Figure 2-4).

Several commercially-available frameworks have been developed that enable heterogeneous tools to be linked together to enable design decision making including ModelCenter, iSIGHT, and AML. Analysis codes can be wrapped and linked either manually or through limited automated capabilities.

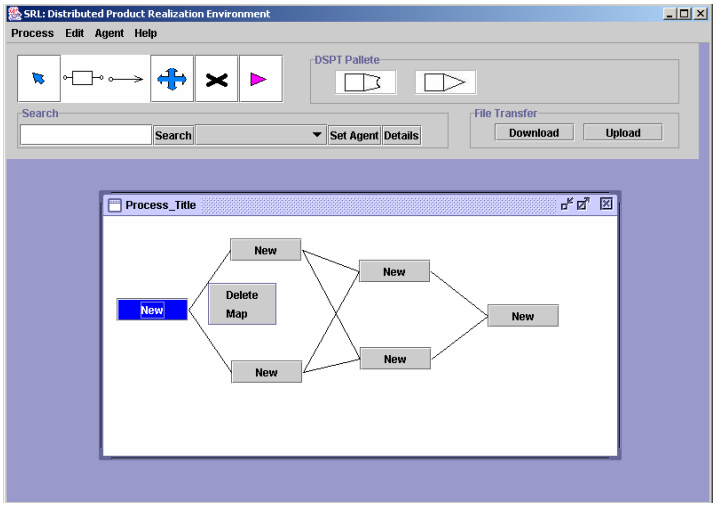


Figure 2-4: X-DPR framework

For example, ModelCenter is an environment which enables software tools and programs to be published in a distributed network. There are several components in the Phoenix Integration tool suite including Analysis Server and ModelCenter. Analysis server is a repository for storing and publishing models across extended networks. Analysis server is the foundation on which complex engineering processes and decisions can be modeled. Analysis server uses the HTTP protocol to share models. Analysis server enables models to be wrapped based on input and output, such that it can be accessed and used throughout the enterprise. For example, FEA models can be developed in ANSYS, wrapped with Analysis Server protocol and published and used throughout the extended design team. Analysis server provides the ability to use standard wrapper or develop customized wrappers. ModelCenter is visual development environment for representing processes. ModelCenter enables several models to be executed at once and data to be shared between those models. While any engineering process can be represented in ModelCenter, it is especially useful for modeling multi-objective engineering design decisions that rely on distributed models. For example, several analysis models may be deployed and linked together in ModelCenter to represent a compromise decision support problem (see Figure 2-5).

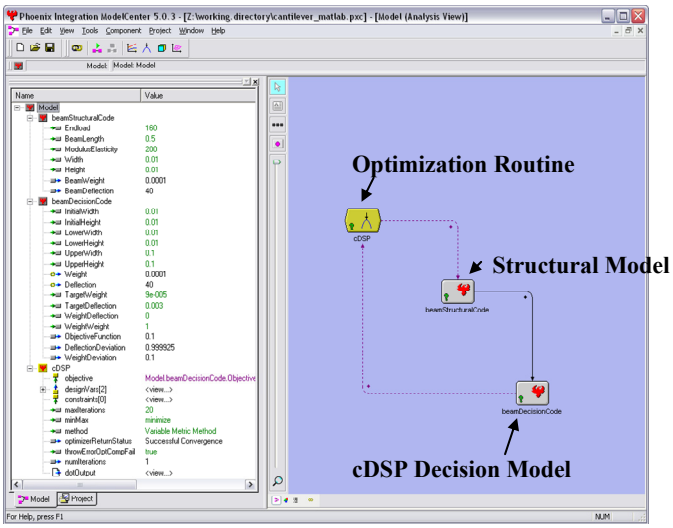
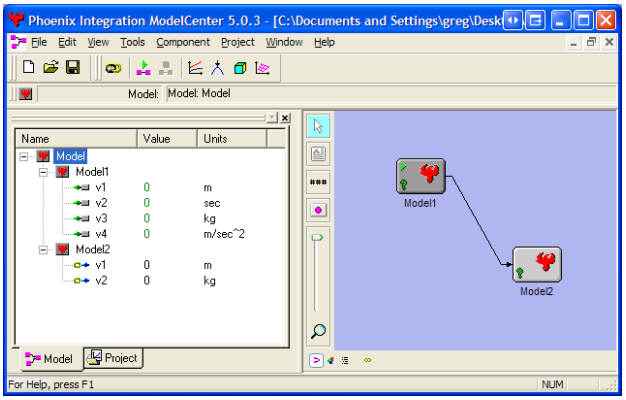
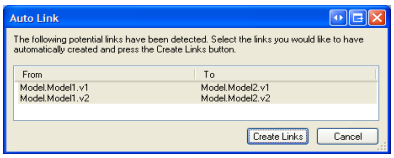


Figure 2-5: Integrating multiple models in ModelCenter

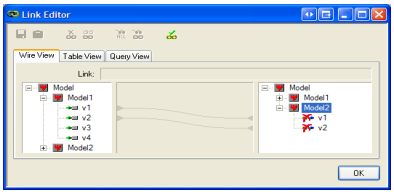
ModelCenter, for example, provides the automated linkages based on syntactic equivalence (see Figure 2-6).



(a) ModelCenter development environment



(b) Automatic linkage



(c) Linkages between variables

Figure 2-6: ModelCenter as a development environment for connecting multiple models in multi-disciplinary design decision making.

For example in Figure 2-8b, the linkages created between Model1.v1 and Model2.v1 and Model1.v2 and Model2.v2 are based on the fact that the variable names are the same.

The units between the v2 in each of the models are not consistent, but the linkages between the variables are valid. However, as a design problem or the analysis models used to support an engineering design decision vary, the linkages between the models and the decision constructs are likely to change. The Launch Vehicle Language (LVL) data model is established to provide a common vocabulary for launch vehicle data between analysts. The LVL enabled a higher level of collaboration through a unified data model and enabled the ability to define interfaces between engineering analysis capabilities and domains. The LVLLinker is an algorithm based on the LVL and the ModelCenter environment to enable the development of linkages between analysis tools in a more flexible manner. The LVLLinker addresses a fundamental problem in multi-objective decision making of linking tools together and sharing data.

Zweber and co-authors [171] utilize the Adaptive Modeling Language (AML) as a means for enabling communication in the design and analysis of an aircraft wing. The AML is an object-oriented knowledge representation framework that enables designers to capture knowledge from the multiple domains and create parametric models . [6; 153]. The AML environment enables models to be developed and instantiated with different values as the design changes AML provides a flexible development environment to develop models that can be used and tailored in different applications. AML provides classes, functions, and methods for geometric models, automated meshes, multi-physics analysis integration, and optimization to name a few (see Figure 2-7).

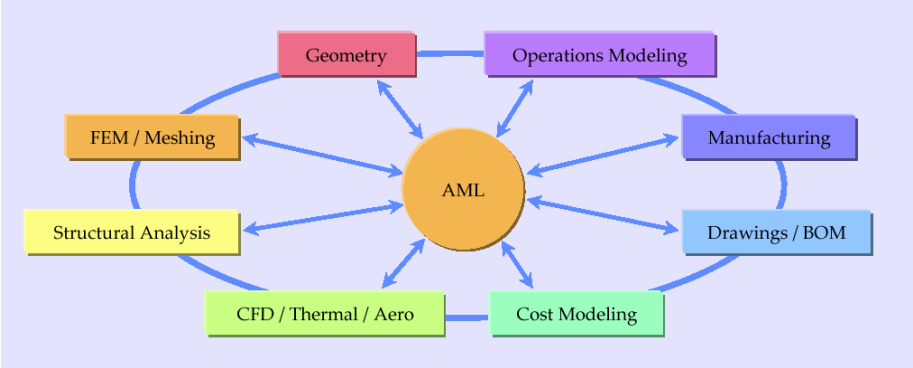


Figure 2-7: AML common computational model [6]

Finally, optimization modeling languages are proposed to bridge the gaps between model formulation and optimization solution techniques [52]. Optimization modeling languages are computer-based modeling languages that support the formulation of optimization problems and subsequently translate the problems into a mathematical representation. Optimization modeling languages provide a clear and concise mechanism for describing optimization problems. One such modeling language is the Algebraic Modeling Language. Optimization problems are represented in terms of sets, indices, parameters, variables, and constraints. Additionally, the Algebraic Modeling Language relies on declarative statements, thus reducing the number of statements needed and the need to specify input and output variables. The primary goal of optimization modeling languages is to enable decisions makers to formulate optimization problems.

2.6 Design and Analysis Integration for Engineering Design Decisions

Design decision making is the process of maximizing or minimizing an objective while satisfying system constraints and bounds [19]. An essential component in design decision making is determining or predicting the behavior of the system throughout the feasible design space, such that the value of the objective function can be computed. In

engineering design, the behavior of the system may be determined through physical prototypes and tests, through first principles, or through the use of computer simulations and complex analysis code. For example, computational techniques such as finite element analysis (FEA) for structural analysis or computational fluid dynamics (CFD) for fluid mechanics may be used to predict the behavior of the system. These predict the behavior of a virtual prototype with the aims of achieving a high correlation between predicted behavior and actual system behavior. In the context of multi-physics simulations, a combination of numerical techniques may be used to predict the overall system behavior. For example, to predict the behavior and performance of an aircraft wing, designers may need to couple computational fluid mechanics and structural analysis. Thus, the integration between design representations and analysis models in the context of engineering decisions is required in the design of complex systems.

There are several issues associated with integrating the domains of engineering design and analysis that include: (1) syntactic issues - the disparity and interoperability in design and analysis tools and (2) semantic issues - context-dependent product representation, idealizations and simplifications between engineering design and analysis models [46]. Researchers have addressed the integration of design and analysis from a broad base including standards-based product representations [57; 65], design repositories for function and behavior based design [22-24; 97; 145; 147; 148], automatic mesh generation and shape modification [15; 134; 135; 150; 151], attribution and feature recognition of product models [13; 14; 68], and model idealization and simplification to name a few [50; 51; 118; 136; 137; 161]. While there has been more than a decade of

research effort and technology development, there are still many opportunistic areas for advancement.

As previously stated, Design-analysis integration (DAI) is the research area that addresses the need to share information between design and analysis domains in a computational means. In this context, design is used synonymously with the specification of product geometry through the use of computer-aided design (CAD) software and analysis refers to computational tools for simulating product behavior. A core issue associated with product development is the gap between engineering designers and analysts [46]. To illustrate their point several interaction scenarios are presented, these include:

- *Retroactive Analysis Scenario* - the artifact is designed based on the designer's knowledge. The design is then validated at the completion of detailed analysis by analysts. Often times, Retroactive Analysis results in over design and long realization cycle times.
- *Integrated Spatial-Functional Design* - Design decisions are supported by analysis knowledge throughout the major phases of the realization process. The design is analyzed at the completion of each phase during product development. The artifact is analyzed at varying levels of detail from conceptual design to detail design to support engineering decisions. The integrated approach relies on analysis to be performed at various levels of detail throughout the design process from conceptual level analysis and design to detailed analysis and design.

In these scenarios, analysis is performed to support engineering design decisions. A key difference between the retroactive and integrated scenarios is the frequency at which

analysis is completed during product development. In the Integrated Spatial and Functional design scenario, the behavior of the product is simulated more frequently, thus enabling designers to explore the design space and verify the behavior of the product more readily. The common thread in each scenario is the knowledge shared between product design and analysis. Unfortunately, knowledge exchange between design and analysis may be inhibited due to several reasons including heterogeneous tools and different representations between design disciplines. A generalized model of these scenarios is presented in Figure 2-8.

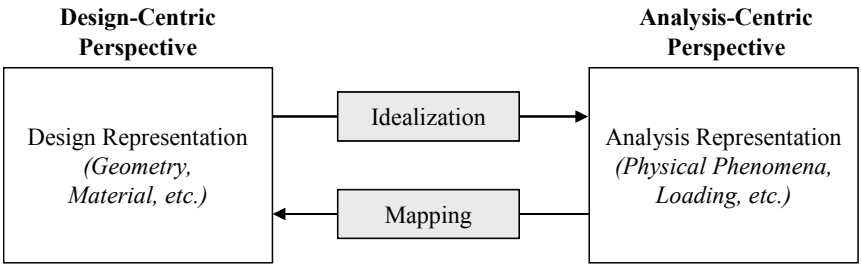


Figure 2-8: Integration of design and analysis activities.

The relationships between design and analysis perspectives are captured as knowledge intensive mappings or idealizations. In reality the *Idealization* and *Mapping* relationships between design and analysis may be represented as complex rules of thumb, algorithms, or data translations. For example, the detailed product geometry specified in the design-centric perspectives may be simplified using expert systems based on the results of interest and knowledge about the analysis process. To address this problem, and thereby reduce the gaps between design and analysis activities, it is necessary to capture and share knowledge across the design perspectives. There have been several research thrusts in the area of design-analysis integration. Current research efforts have focused on capturing geometry and geometric relationships between design and analysis, but have

failed to capture decision-related information. The implicit assumption in DAI is motivated by the need to shared *product geometry between engineering design and analysis applications.*

Computational frameworks have been developed to realize various aspects of DAI. For example, the multi-representation architecture (MRA) is a conceptual framework proposed by Peak and colleagues. The MRA attempts to bridge the gap between traditional design (CAD) and analysis (CAE) tools while satisfying the need to link CAD and CAE in a traditional routing analysis [116]. The MRA is based on the following four building block constructs: (1) Solution Method Models (SMM), (2) Analysis Building Blocks (ABB), (3) product models (PM), and (4) Context-Based Analysis Models (CBAMs). The MRA supports capturing knowledge and expertise for routine analyses through semantically-rich information models and the explicit associations between design and analysis models. While the MRA captures routine analysis and the mapping between design parameters and analysis parameters, there is still the opportunity for misuse of the analysis templates. The assumptions, variable definitions, and application context are not explicitly captured in the context of engineering design decisions.

The conceptual architecture proposed in the MRA, is computationally realized through the development of the constrained objects (COBs) knowledge representation [165]. The COB representation is based on objects and constraint graph concepts. Non-causal constraints are used to represent relationships in COBs because a variety of inputs and outputs can be specified. In other words, a single constraint can be used to represent different input/output flows in engineering systems. There are several representations of COBs that include a text-based modeling language. The lexical form is computer-

processible while the graphical language is human interpretable. COBs are a non-declarative representation that is a combination of object and constraint graph techniques. The COBs representation supports superclass/subclass concepts, non-causal representations, and relationships objects.

STEP provides a standardized mechanism for exchanging information between heterogeneous design support software. Although it was initially intended to facilitate the exchange of data between disparate CAD software, STEP has grown to provide additional support and information exchange between diverse software applications. Specifically, STEP standards have been developed to address the DAI issues. STEP-AP209, entitled Composite and Metallic Structural Analysis and Related Design, provides standardized information models for sharing product information between design and analysis software. Primarily, AP209 provides a mechanism for capturing product geometry finite element analysis models, controls, results, and analysis reports [7]. The conceptual architecture of the AP209 information model is presented in Figure 2-9.

STEP AP209 addresses interoperability of product models between CAD and FEA applications, thus enabling closer integration of design models and analysis models. While AP209 addresses interoperability issues between diverse software tools, it does not capture the knowledge-intensive approach of idealizations between models. For example, AP209 provides a mechanism for relating *Nominal Product Geometry* to be related to *Idealized Analysis Geometry*. However, AP209 serves primarily addresses this through the notion of *packaging* relevant information between design and analysis together. The standard does not explicitly capture the complex relationships, such as idealizations,

simplifications, and abstractions of product-related information between design and analysis. In this context, the architecture developed by Peak is superior by providing the capability to capture complex relationships between design and analysis models.

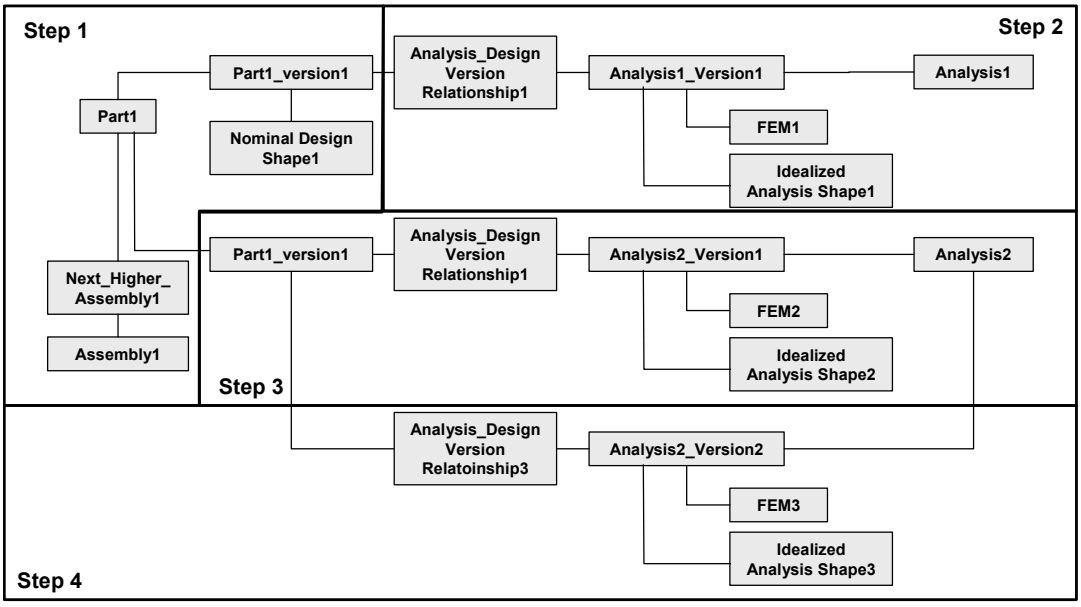


Figure 2-9: Conceptual architecture of AP209

AP209 is a useful standard, but is limited in scope and reuse. The next generation of STEP is moving towards modularization of engineering information. The Engineering Analysis Core Model (EACM) has been proposed as the standard information representation for engineering analysis. The ECAM is an emerging ISO standard that is part of STEP modules. The EACM bridges the CAD and PDM with project management and leading edge analyses applications. PDM was first concerned with the structure of the product. PDM was a good way to provide an index of engineering information and to track dependencies. PDM is a key component to enable engineering analysts to keep track of analysis models and the related design models, as well as analysis results. Two major problems in archiving information is the format of the information and the

semantics, or meaning, of the information. PDM in the context of engineering analysis is concerned with the product, the activity of the product, and the state of the product. Changes in the state of the product are as important as changes in the product itself.

The DAIM is proposed for capturing tighter integration between design and analysis [45]. The DAIM provide the ability to capture relationships between design and analysis perspectives through the Common Core Relationship. In this context, the DAIM enhances the *richness* of the relationships captured between engineering design and analysis. For example, the Common Core Relationship can be used as a container to capture simplification and idealization rules between design and analysis. The DAIM model provides a bridge between product information modeling and artificial intelligence for design analysis integration. For example, the Common Core Relationship can be used to capture idealization and simplification in design [15; 50; 51; 134; 136; 137; 161].

2.7 Multiple Views in Engineering Product Realization

Many researchers are currently exploring design and analysis integration through the concept of a hierarchical modeling paradigm. The general concept of the hierarchical modeling paradigm is product data for the entire life cycle of a product is stored as a master model in a design database. The product data can be viewed through different domain-specific aspects or views by applications to generate, display, or modify the product information. The master model serves as the global schema for all product data and the analysis models serves as user views of the master model. Based on this concept, the product model remains complete and comprehensive, because it holds the most up-to-date product data.

Hoffman and Joan-Arinyo [62; 63] present a plausible organization for a product master model. The authors develop an architecture that keeps consistent associations between CAD and downstream applications. The architecture accounts for associating product data between various applications, while maintaining the proprietary data of the applications. Design-analysis association is predicated on the ideas that the master model is an object-oriented repository that has mechanisms for maintaining the integrity and consistency of the information structures for the various engineering domains. The master model has several clients, one of which is the CAD application. Additional clients associated with the master model may deal with manufacturing process planning, geometric dimensioning and tolerancing, cost estimation, etc. Change protocols enable downstream application to implement its own semantics and provide intimate knowledge of the analysis domain. The change protocol eliminates the burden to create custom associations for each different CAD system.

Yoshioka and Tomiyama [169] present a mechanism for integrating various aspect models, such as geometric models, kinematic models, and finite element models for knowledge intensive engineering. The Knowledge Intensive Engineering Framework (KIEF) is constructed using multiple objects (i.e., aspect models) expressed through a metamodel mechanism. The metamodel represents the relationships between the concepts in the aspect model. The framework proposed aims to integrate and maintain the consistency of the various models in all of the product phases. The KIEF framework integrates commercially available software tools through the idea of the "Pluggable Metamodel Mechanism". An aspect model is a model of a designed artifact from a particular point of view. For example a finite element aspect model may be completely

different that the geometric shape aspect model. The metamodel mechanism provides the framework for integrating the many aspect models associated with a technical artifact. Physical concepts and mechanical components are captured in the metamodel mechanism. The metamodel mechanism describes how information is exchanged among the aspect models. However, it is not always easy to extract all the necessary parameters to complete the aspect model. For this reason, the ability to plug in existing modelers is presented. De Martino and co-authors [40] present an approach to the integration of the design of an engineering artifact (CAD) and the downstream engineering processes (CAE) based on feature based modeling using design-by-features and feature recognition. To achieve the integration between design and engineering processes, a common model must exist. The shared model provides different views for different analysis domains. Toward this end, the intermediate model (IM) is developed. The IM is shared between applications and provides them with context-specific feature-based views. Initially, the designer creates an object using design features from a library. The design is evaluated and stored in the IM to maintain the semantics of the features. Based on the stored data, the feature recognition is stored for each of the analysis domains. An integrated feature-based system is developed to link design and engineering activities. The IM links the parametric model and the shape model. Additional semantic information allows for application specific views for various engineering contexts. The intermediate modeling process provides the main source of information for various processes.

Bronsvoort and Noort [30] conduct a comprehensive literature survey in multi-view modeling and summarize shortcomings in the previous described multi-view modeling approaches as the focus on later stages in design where product geometry is the focus and

a one-way association from design to different views. To meet these needs, a framework is presented that enables product features to be captured at various stages of the design process and enables two way flows of information between different views. Additionally, the authors focus on four design phases: conceptual design, assembly design, part detail design, and manufacturing design. In conceptual design the functional components of design are addressed without having to completely specify the geometry of the part. In assembly design, the associations between components and the basic geometry of components are specified. In part design, the detailed shapes of the parts are specified. Finally, in the manufacturing design the manufacturing processes are specified. In the design process, each phase has a particular view of the product that are defined through classes of features. Feature models are checked for validity and consistency with other feature models in the process.

Multi-view / aspect models present a plausible means for designing a product from the top-down or bottom-up approach given a significant and relevant number of features can be defined a priori to the design process. The multi-view approaches reviewed enables product information from diverse software applications to be “linked” and changes to be managed throughout the development process. However, the approaches fail to address *why the multiples views are linked.* The inherent shortcoming of multiple-view approaches, similar to standardized product models, is the focus on product geometry and inability to capture decision making information.

2.8 Discussion: How are the Constructs used in this Dissertation?

Decision-centric design provides the context and "big picture" view of the engineering product realization process. Decision-centric design is the philosophy of

representing design as a set of inter-related decisions in which information is integrated from several disciplines throughout the product realization process. Decision-centric design is realized through multi-objective decision constructs. Multi-objective decision constructs, specifically the cDSP, provide a mathematical formulation for modeling decision commonly encountered in design. The cDSP is a domain independent construct that enables designers to model decision to determine the "right" values (or combination) of design variables (e.g., system parameters), such that, the system is feasible with respect to constraints, preferences, bounds, and goals, and that system performance is maximized. The cDSP can be used as a means for packaging relevant decision-making information and sharing it with stakeholders in the product realization process. However, the cDSP (and other design decision formulations) have kept pace with information and data exchange in engineering design. Product data exchange has received significant attention and lays the foundation for developing information models for enable the exchange of digital product information. However, product information models have not been developed to adequately address multi-objective decision making. Product information exchange provides the basis for advance representation of design decisions. Additionally, frameworks for supporting multi-objective design decisions and the integration of design and analysis are becoming increasingly important. However, a missing component in multi-objective decision frameworks is the underlying information modeling representations. Conversely, design analysis integration technologies have failed to take into consideration multi-objective design decisions. Finally, multi-view approaches have provided several technologies and techniques for linking "views" of the product. The multi-aspect view approaches have laid the groundwork for integrating

product information from different perspectives. A summary of the foundational constructs and their usage in this dissertation is presented in Table 2-4.

Table 2-4: Summary of constructs and their usage in this dissertation

Construct	Use in this Dissertation
Decision-centric design and the Decision Support Problem Technique	• A philosophy for modeling engineering design as a set of decisions. • Provide the high level picture and motivation for pursuing the research • Strategy for integrating information from multiple perspectives • Provide the basis for formulating research hypotheses for integrating multiple perspectives
Multi-objective decision making and the Compromise Decision Support Problem	• Mathematical construct for realizing the integration of disciplinary analysis models • Provides the foundation on which information representations are developed • Highlight gaps in current frameworks and the need to represent and capture design decision in a structured manner
Product information exchange	• Need for exchanging information in design • Highlights current focus of PDE • Foundational for developing explicit representation in design

Table 2-4: Summary of constructs and their usage in this dissertation (continued)

Decision frameworks	• Provides the need for information model of engineering design decisions. • A current gap in decision frameworks is the lack of computational representations of decision formulations
Design analysis integration	• Establish the need to exchange design and analysis information • Provides the foundation for developing representations
Multi-view models	• Approach for integrating multiple aspects in engineering design • Motivation for pursuing a decision-centric perspective

The purpose of completing the literature survey is to identify gaps in constructs and develop research questions. Thus, in the following section a detailed discussion and critical review of the constructs and current state of the art is presented with the motivation for identifying research opportunities.

2.9 Summary of Gaps and Research Opportunities

The primary purpose of the literature review is to identify gaps and research opportunities in the area of engineering information for design decisions that are relevant to the focus of this dissertation. Considering the critical review of relevant literature, the following issues have received little attention:

- A systematic method for formulating multi-objective design decisions and integrating disciplinary analysis models is lacking. The formulation of engineering design decisions is largely an art form. Current MDO frameworks provide low level support

for integrating computer code and linking analysis packages, but do very little to address the formulation of design decisions at a higher level.

- Decision constructs provide high level, natural language structure for design decisions. Models for representing engineering design decisions are at two ends of the spectrum. On one end, optimization modeling languages have been proposed for representing the mathematics of engineering design decision. On the other end, word formulations of engineering decision constructs have been developed. However, the semantics of engineering design decision is often lost. Current decision support frameworks are limited in the information representations that they capture.

- Current information models for engineering design focus on product representations. Product models are in largely limited to the representation of product form. The representation of process information has been proposed by several researchers but has not been integrated with product model. There is a need to model the engineering decision taken throughout the design process that integrates product information and process information.

- Engineering design decisions are currently represented as a flat structure. The complex relationships between decision information are not captured. For example the relationships between analysis support models and constraint equations are not captured in the cDSP formulation. Additionally, the relationships between design parameters and analysis parameters are not established. There is a need to develop a richer representation of design decisions such that information can be reused across multiple decisions.

- Decision-centric design has been proposed as a "language" for representing engineering design processes. However, the formalization of the language, in a computational sense, has not come to fruition. The implementation of decision centric design in a computational environment is limited.
- Similarly, a computational based representation of the information associated with engineering design decisions is needed to share decision information across the extended enterprise. Engineering decision making has been proposed as a common "language" for design however the computation-based representations have not kept pace with other developments.

2.10 Chapter Synopsis

The primary role of the discussions in this chapter has been to justify the research questions and hypotheses proposed in this dissertation by establishing their originality, significance, and theoretical structural validity in the context of relevant literature. In this chapter, the relevant literature has been reviewed critically in the areas of decision-centric design, multi-objective decision formulations and frameworks, information exchange in engineering design (see Figure 2-10). The next step is to discuss the implementation of the proposed information model in Chapter 3. In Chapter 3, the underlying information modeling formalisms are presented and assessed in the context of requirements for engineering information management. The aims in this chapter are addressed:

- ✓ To introduce and critically evaluate relevant literature – The current state of the art is presented and critically evaluated in the domains of engineering information modeling, design analysis integration, and multi-objective design decision making.

- ✓ To identify the gaps in current approaches – Several gaps are identified based on the critical review of literature and research questions are formulated.
- ✓ To establish the research opportunities – The research opportunities are identified based on a discussion of current literature and available constructs.

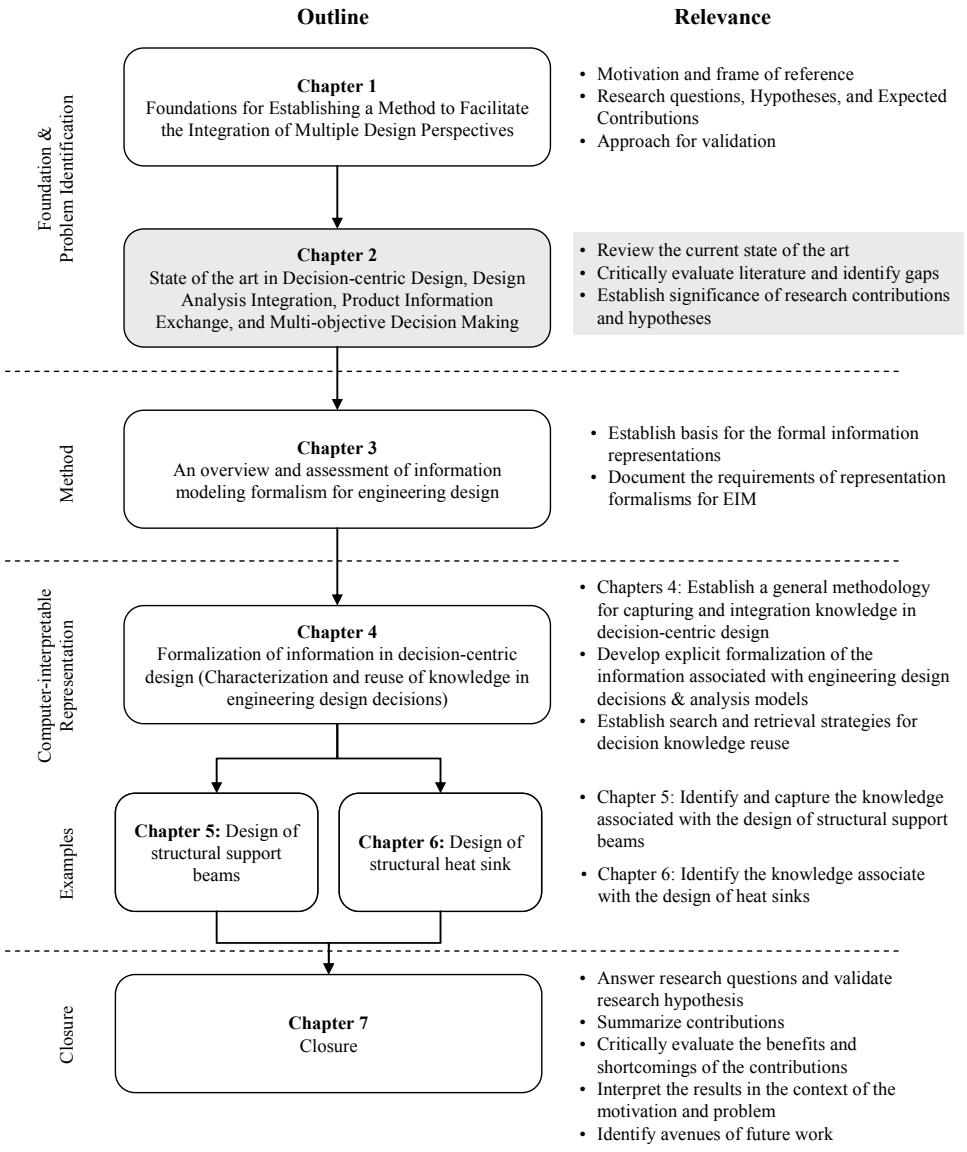


Figure 2-10: Outline of dissertation

CHAPTER 3:

INFORMATION MODELING IN ENGINEERING DESIGN

Aims

- To establish the motivation for developing a formalism for design decisions.
- To introduce the information modeling requirements for modeling design decisions.
- To critically assess modeling formalisms for capturing information.
- To evaluate DL for addressing engineering design problems.

In this chapter, the second research question and hypothesis are partially addressed. The second research question is *"How can the information and knowledge associated with the cDSP and analysis support models be represented in a computational environment?"* Recently, description logic (DL) has received significant attention in current literature as a representational formalism for developing engineering information models. However, the motivation for using DL over other formalisms for EIM is not established. Thus, in this chapter, three common information modeling formalisms, including the semantic data model, object-oriented data model, and description logic, are critically assessed. First, characteristics of engineering information management problems are identified and requirements are extracted. The modeling formalisms are evaluated in light of the EIM characteristics and requirements. This chapter concludes by recommending DL as an appropriate modeling formalism for engineering information management. Specifically DL is recommended because of the advantages of information consistency, organization, and extensibility.

3.1 Overview: Information Modeling

Information modeling is the process of specifying the structure of information used within an application. Information modeling has been the subject of several fields including database development, information systems, software engineering, knowledge representations and programming languages [33; 88]. Information modeling is concerned with the specification of symbols that model a domain of interests in a computational-processible form. In this dissertation, information modeling and information bases are used to refer to databases and knowledge bases [100]. There are several formalisms for developing information models including semantic models, object-oriented models, and logics-based approaches. These formalisms enable classes and instances of the classes (individuals) to be declared using an established syntax and semantics. In this work, we review approaches for explicitly representing engineering information. In the following section an overview of several information modeling formalisms, including semantic data model, is presented.

3.2 Motivation for Information Management in Engineering Design

As previously stated, decomposing a system into subsystems often results in inefficiencies and difficulties in the communication and integration of product information between designers [80]. For example, interoperability and information exchange problems in the automotive supply chain are estimated to cost nearly $1 billion U.S. dollars per year [31]. The need for engineering information management (EIM) systems and technologies is motivated by: (1) the immense amount of digital information generated in design and (2) the need to share the information across the extended enterprise. EIM technologies have both helped and hindered information capture and

exchange. Existing software tools do not address information exchange and coordination, but rather increase problems by isolating product information at the boundaries of the specific tools, resulting in a *Tower of Babel* [61]. In the following sections, the development of engineering information models is presented.

3.3 Framework for Developing Information Models and Ontologies

The development of information models and ontologies is as much a product as it is a process. The outcome of formalizing the information associated with engineering design decisions comprises: (1) a clearly defined domain of discourse and scope, (2) identification of reasoning and organization services, and (3) the conceptual information models and form representation. The representation is only considered valid when these three facets are taken as a whole. For example, the scope and domain of discourse of the information models can easily be "broken" by extended the querying requirements. Similarly, the reasoning and query services may not be consistent if the representation is modified. Hence, a systematic approach for capturing the information associated with decision-centric design must be followed. The information modeling method used in this research is based on [60; 104; 156]. The process is illustrated in Figure 3-1.

In this chapter we are primarily concerned with assessing information modeling formalisms for engineering information management. In this context, characteristics of engineering design problems are identified and high-level requirements are developed. Based on these requirements several information modeling formalisms are evaluated. In the following sections three common information modeling formalisms are discussed.

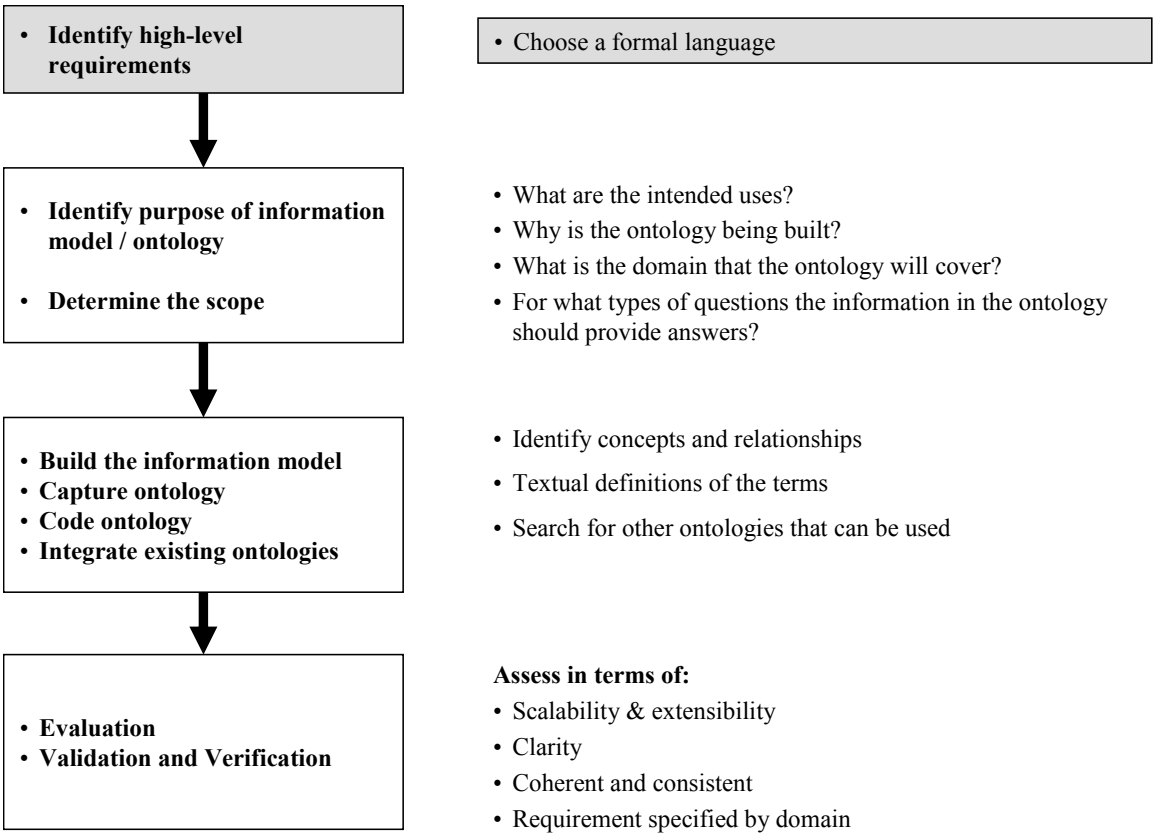


Figure 3-1: Information model and ontology development process

The assessment of the underlying information modeling formalism is motivated by the myriad of product models discussed in Section 2.3. While the issue of information exchange in design has been investigated extensively there is not a *grand unified information model* that can support all phases of engineering design. As a result, there are nearly as many engineering information models as there are information modelers, and the number of models keeps growing. Thus, the primary goal in this chapter is not to propose another information model, but rather to assess modeling formalisms for developing information models.

Recently, there has been a growing interest in the use of description logic (DL) for engineering information models. For example, semantic web technologies and ontologies

have most recently been utilized for capturing engineering information. Researches have used the web ontology language (OWL) for engineering information that can be shared over extended networks [9; 78; 101; 111]. While, McGuinness and Wright [88] provide several criteria for assessing the value of DL for configuration modeling in telecommunications, as a whole current literature has not rigorously assessed the underlying modeling formalisms for engineering information management and the advantages of DL over other modeling formalisms.. Thus, in this chapter research question and hypothesis two are addressed. In this chapter, we build on the foundational work by Bordiga [26], Calvanese [33], and Baader [16] on the use of DL for information modeling and the work by Eastman and coauthors [41] for developing engineering information models. The primary objective in this research is to assess formalisms for engineering information management, not to develop information models for a particular engineering domain.

3.4 Characteristics of Information Management in Engineering Design Problems

In this research a set of characteristics are identified for information management in engineering design problems. Below is a list of characteristics about engineering design problems as related to engineering information management.

- **C1: Different terminology is often used in the design process** [60; 61]. In the design of complex, multidisciplinary systems, it is unreasonable to expect that all designer will share a common vocabulary. Engineering design may use different terminology to describe shared concepts across disciplines. This adds a level of complexity associated with information exchange [144].

- **C2: Design problems are often addressed from multiple levels of abstraction and detail** [40; 43; 46; 127]. Engineering design problems are often addressed at varying levels of detail. For example, low-fidelity analysis models may be used to predict the convective coefficient or a high-fidelity computational fluid dynamics (CFD) model may be used to compute the convective coefficient [114]. In each of these approaches a different set of product information may be of greater importance.

- **C3: Complex engineering systems are organized hierarchically** [76; 77; 105; 140]. Engineering systems are hierarchical in nature including assembly-component, requirements allocation, systems level and component level optimization, and multi-scale modeling. In the fin array heat design problem, several analysis models are used at varying fidelities. For example, complex systems are often decomposed hierarchically into components or sub-systems, thus enabling the designer to address more manageable problems and even identify problems that are not readily apparent from a holistic perspective. Additionally, designs are often analyzed in terms of function-subfunction hierarchy.

- **C4: Information is dynamic throughout the design process** [61; 138]. The product information associated with design is constantly changing from the conceptual stages of design to the detail stages. Additionally, design is an iterative process in which digital product information is created and modified at a high rate [144]. These changes not only include changes associated with specific instances of design information, but the structure and organization of design information is also varied. For example, the information structure of a design decision may change according to the analysis models used to support the decision.

- **C5: Design knowledge is often reused** [23; 98; 124; 146; 149; 159]. Engineering design reuse exists at the physical and virtual level. Designer reuse standardized parts to reduce the cost of manufacturing and maintenance, reuse parametric CAD models for variant and scaled design representation, and reuse analysis and simulation models to determine the behavior of systems.
- **C6: Engineering design information has complex data structure from heterogeneous tools** [4; 65; 87]. Modern engineering products are typified by complex structures and interrelationships between components, multiple functions, and tailoring of materials properties. As a result, the information models for engineering design are often cooperatively developed by teams.

These characteristics provide the basis for developing engineering information management systems and the modeling formalism required for accurately representing engineering information.

3.5 Requirements for developing an information model for capturing the semantics in engineering design decisions

There are several requirements associated with the developing an information model for decision-centric design. An initial list of requirements is provided in Table 1-2. The purpose of the information model is to provide support for explicitly representing engineering design information. Several requirements are presented that define the scale and scope of the information model. The requirements are derived based on the existing literature [67; 72; 112; 129; 164]. The requirements are divided into two sets. The first set, discussed in this chapter, addresses the underlying formalism used for developing the

information model. This set of requirements is at the meta-level. The second set of requirements, presented in Chapter 4, is used to define the scope of decision related design information captured.

3.5.1 Requirements for Information Modeling Formalism

- **R1. The information model should be extensible.** The information model should be defined such that decision-related knowledge from a variety of discipline can be represented within the scope and bound of model. For example, the information model robust to enable the representation of analysis models not initially considered during the initial development. Engineering decision makers should be able to represent new and innovative analysis models and design decisions from multiple design perspectives.

- **R2. The information model should enable consistency checking of concepts.** The information models should be able to detect inconsistencies of the concepts defined in the representation. This is important to ensure the knowledge and information shared between disciplinary experts is correct. It is essential that the representation of design decisions analysis models are free of error.

- **R3. The information model should provide information organization capabilities**: Engineering information models should support hierarchical organization of design information. As previously stated engineering systems are often decomposed in a hierarchical fashion. Thus, engineering information models should support common used engineering relationships including part-of, is-a, and specialization.

In the following section, the design characteristics and information modeling requirements are correlated to ensure that the requirements address the needs of engineering design problems.

3.5.2 Design Characteristics and Information Modeling Formalisms

The design problems characteristics are correlated against the information modeling requirement in Table 3-1.

Table 3-1: Correlation of design characteristics and information modeling requirements

EIM Characteristic	**EIM Requirement**		
	R1: Extensible	**R2:** Consistent	**R3:** Organization
C1: Differing terminology between designers	●	●	○
C2: Multiple level of abstraction	●	●	●
C3: Hierarchical information organization		○	●
C4: Dynamic information models			●
C5: Reuse of design information	●	●	●
C6: Heterogeneous design support tools	●	●	●
Key: ○ - Weakly related; ● - Strongly related			

Extensibility (R1) - It is important to be able to quickly and easily extend the scope and the concepts in the information to enable communication between different design disciplines as needed. However, as new concepts are added to the information model, it is important to ensure the existing concepts are maintained.

Information consistency (R2) - It is important in developing complex engineering information models to ensure the concepts and relationships between concepts are correctly modeled and consistent. For example, STEP APs have recently been revised for consistency across different APs and modularization. As a result significant time and effort has been allocated to identifying inconsistencies. Complex information models may be created by several developers and mechanisms must be in place to check the internal consistency of the schema including: syntactically identical concepts, semantically equivalent concepts, and inconsistent concept definitions.

Information organization capabilities (R3) - As previously stated engineering systems and the associated information are often decomposed and organized in a hierarchical fashion. Additionally, designers often reuse information across several different design problems. Thus, the information models should provide a means for organizing and retrieving design information in hierarchical taxonomies. As new information and modifications are made to the information model, the organization should be updated. Finally, information models should support information reuse by enabling designers to retrieve stored information through query and retrieval services.

3.6 Information Modeling and Knowledge Representation Formalisms

There are several formalisms for developing information models including semantic models, object-oriented models, and logics-based approaches. These formalisms enable classes and instances of the classes (individuals) to be declared using an established syntax and semantics. In this work, we review approaches for explicitly representing engineering information. In the following section an overview of several information modeling formalisms, including semantic data model, is presented.

3.6.1 Semantic Data Model

Semantic data models (SDM) are primarily used in the development of relational database schema. There are several techniques for modeling a domain with semantic data models including the Entity-Relationship and Unified Modeling Language (UML) class diagrams. The Entity-Relationship (ER) model developed by Chen [35] is one of the most popular methods for developing relational database schema. The ER model provides a graphical method for explicitly specifying entities, attributes, and relationships in a real world domain. An entity is an object that exists in the domain of interest. A specific object is called an instance of an entity. An entity may have several attributes that further describe that entity. Attributes are the smallest level of information granularity and can be represented by basic types such as `Boolean`, `real`, and `integer`.

Several entities may be related to each other through relationships. A relationship denotes an association between instances that participate in the relationship. Additionally, the degree of the relationship can be specified as unary, binary, or ternary. Relationships between entities are often constrained by the cardinality and the participation role. Cardinality defines the number of instances that a relationship can participate in such as one-to-one (1:1), one-to-many (1:N), or many-to-many (N:M). Relationships can be specified to represent a variety of associations between concepts; common relationships include `is_a` and `part_of` relationship to create hierarchical taxonomies. ER schemas can be translated into a implementation-level logical schema for relational databases using established mapping rules [102]. An illustrative example is presented in Figure 3-2.

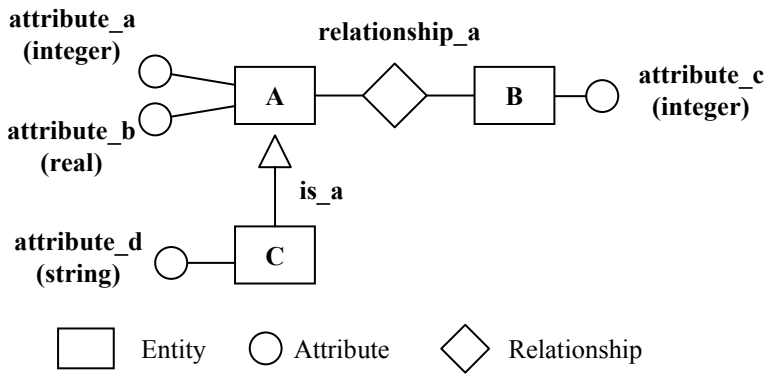


Figure 3-2: ER schema for simple example

The schema consists of three entities (things): `A`, `B`, and `C`. Entity A has two attributes: `attribute_a` is an integer value and `attribute_b` is a real value. Entity `B` is defined to have `attribute_c`. Entities `A` and `B` are associated through `relationship_a`. Finally, entity `C` is specified to be a subclass of `A`. Thus, entity C has the two attributes inherited from `A` and `attribute_d` that takes a string value. A detailed overview of ER modeling is included in [102]. A comparison of ER concepts to DL is presented in [16] and [33].

3.6.2 Object-Oriented Data Model

Object-oriented data models (OODM) were initially proposed as a means for integrating database formalisms with object-oriented programming [33]. In object-oriented programming, classes are created that define the data type, data structure, and functions that can be applied to that data. Additionally, relationships between objects can be created with other objects. An advantage of object-oriented programming over traditional programming is the ability to create reusable objects through super-typing and sub-typing. However, object-oriented programming provides additional functionality

beyond what is needed for information modeling [16]. For example, object-oriented programming provides the ability to describe dynamic properties of classes and function for classes. Information modeling is primarily focused on representing declarative structural properties of objects. Thus, in discussing object-oriented programming for information modeling, a subset of characteristics is considered. An object-oriented data schema is developed by imposing a set of constraints on the instances of the classes through class declarations. Object oriented modeling emphasizes the following:

- ***Class***: the unit of definition of data and behavior. A class is a thing and is a basic unit of modularity and reuse
- ***Object:*** an instance of a class
- ***Inheritance:*** the basic means for specifying subclasses. Inheritance provides a way to define a (sub)class as a specialization or subtype or extension of a more general class

The schema for the simple example problem is represented as an object-oriented schema in Figure 3-3. Object-oriented data models rely on a "transparent" object identifier to uniquely identify an individual in the database. OODM are often "translated" into relational databases to enable storage. A discussion of the mappings between OODM and DL is presented in [16] and [33].

```
class A type_is
  attribute_a: integer
  attribute_b: real
end

class B type_is
  attribute_c: real
  relationship_a: A
end

class C is_a A type_is
  attribute_d: string
end
```

Figure 3-3: Object-oriented schema for simple example

3.6.3 Description Logic

Description logic (DL) form a subfield of knowledge representation and reasoning (KRR) based on formal logic systems. DL has been researched predominantly in the computer science and artificial intelligence communities. However, DL have received a growing interest in several engineering domains including configuration [88], functional modeling of engineering systems [74], and the development of repositories and ontologies for capturing engineering knowledge [59]. DL are founded on three core tenets:

- The basic building blocks of DL languages include atomic concepts (unary predicates), atomic properties (binary predicates) and individuals
- The power of the language is restricted in that it uses a small set of constructors for building complex concepts and roles
- Implicit knowledge about concepts and individuals can be inferred automatically based on standard reasoning services

DL are a family of logics-based knowledge formalisms that facilitate representation and reasoning about knowledge in a structured manner. DL provide a formal syntax and semantics for describing knowledge within a domain in terms of *concepts* and *properties*

that specific *individuals* must satisfy [106]. DL rely on the following entities to model the knowledge within a particular domain [20]:

- ***Concepts*** (classes of individuals) have two functions: (1) they describe a set of objects and (2) they determine properties of individuals.
- ***Properties*** represent relationships between individuals. Properties are often defined at the concept level, but actually relate individuals of those concepts. Properties describe the restrictions on individuals of concepts.
- ***Individuals*** correspond to particular "objects" in the real world. The main properties of an individual are that it can be distinguished from other individuals.

A knowledge base expressed in DL consists of two primary components, (1) the intensional knowledge and (2) the extensional knowledge. The intensional knowledge relates to the terminology of the knowledge-base and is represented by the terminological box (TBox). The extensional knowledge captures specific individuals in the domain of interest through the assertional box (ABox) [16]. The architecture of DL knowledge bases is illustrated in Figure 3-4.

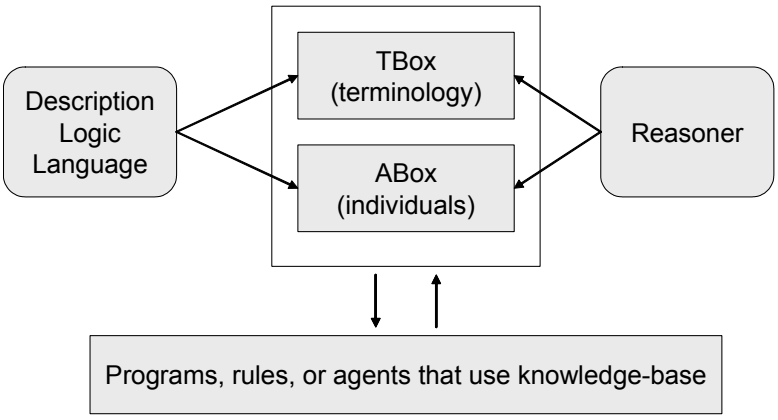


Figure 3-4: Architecture and components of DL [16].

The TBox vocabulary is built from a set of statements about concepts and relationships between concepts (properties). Complex concept descriptions can be defined through logical statements based on atomic concepts, properties, and predefined constructs [46] (see Table 3-2). The ABox captures individuals (instances) of the concepts defined in the TBox. Thus, a DL knowledge base consists of concept terminology and definitions and instances of concepts. The description languages are denoted by the constructs that are utilized in the language. The general notation is $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}]$. DL varies in expressivity, computational tractability and efficiency of reasoning depending on the set of constructs used for the knowledge base. The most basic description language is the attributed language, denoted as $\mathcal{ALC}$. More expressive and increasingly complex languages are extensions of the basic $\mathcal{ALC}$ language. A trade-off must be made between the level of expressivity required to accurately model the domain and support reasoning and the computational efficiency and tractability of the language. However, the expressivity of the DL language chosen for modeling a particular domain cannot be determined *a priori*. In other words, knowing the performance and complexity characteristic of description languages has some influence in the language chosen for modeling a particular domain, but cannot strictly dictate what language is used. The computational tractability, complexity, and completeness of description logic knowledge bases have been the studied extensively [16; 27; 79]. As such, there are well known trade-offs between expressiveness and computational tractability in formal systems. A summary of several description languages and their associated complexities are given in Table 3-3.

Table 3-2: Description Logic constructs

Construct	Syntax	Meaning
Atomic Concept	A	A unary predicate that represents a base concept that has no further decomposition. Atomic concepts provide the base vocabulary
Atomic Property	R	A binary predicate that relates two individuals. The relationships may be object properties or data type properties
Top-most concept	$\top$	The highest level concept in the model
Bottom-most concept	$\bot$	The lowest level concept in the model, no other concept is below
Defined Concept	C, D	A unary predicate that is defined using atomic concepts, properties, and constructs
Intersection (conjunction)	$C \cap D$	The intersection of two concepts
Negation ($\mathcal{C}$)	$\neg C$	Explicit statement of negation for a concept (a concept can either be an atomic concept or a derived concepts
Union (Disjunction) ($\mathcal{U}$)	$C \cup D$	The union of two concepts
Existential Restriction ($\mathcal{E}$)	$\exists R.C$	Describe the concepts (set of individuals) that have at least one specific kind of relationship.
Value Restriction	$\forall R.C$	Describe the concepts (set of individuals) that have at only one specific kind of relationship.
Unqualified at most restriction (N)	$\leq R$	Specifies the maximum number of relationships an individual may have
Unqualified at least restriction (N)	$\geq R$	Specifies the minimum number of relationships an individual may have
Unqualified exactly restriction (N)	$= R$	Specifies the exact number of relationships an individual may have (a combination of at-most and at-least restrictions)
Qualified at most restriction (Q)	$\leq n R.C$	Specifies the maximum number of relationships an individual may have for a particular type. They are more restrictive than unqualified restrictions
Qualified at least restriction (Q)	$\geq n R.C$	Specifies the minimum number of relationships an individual may have for a particular type. They are more restrictive than unqualified restrictions
Qualified exactly restriction (Q)	$=n R.C$	Specifies the exact number of relationships an individual may have for a particular type. They are more restrictive than unqualified restrictions

The description language used in developing the design decision vocabulary and representation is ALCON which has a complexity of PSpace-complete. As a result, with highly optimization and efficient reasoning algorithms, ALCON provide consistent and complete reasoning for satisfiability and subsumption at both the Tbox and Abox levels. A useful resource for determining the complexity of reasoning with DL is [170].

Table 3-3: Description language and complexity [123]

Description Language	Complexity
$\mathcal{FL}$ $\mathcal{AL}$	P
$\mathcal{ALE}$ $\mathcal{ALR}$ $\mathcal{ALER}$ $\mathcal{ALU}$ $\mathcal{ALUN}$	NP
$\mathcal{ALC}$[$\mathcal{O}$]	Pspace
$\mathcal{ALCON}$	Pspace - complete
$\mathcal{SHINQ}$ (simple roles in restrictions)	Pspace
$\mathcal{PDL}$ (Role composition / complement)	EXPTime
$\mathcal{SHIQ}$ with transitive roles	NEXPTime?
KL-ONE	undecidable

The TBox statements for the simple example problem are represented using $\mathcal{ALEN}$ DL in Figure 3-5.

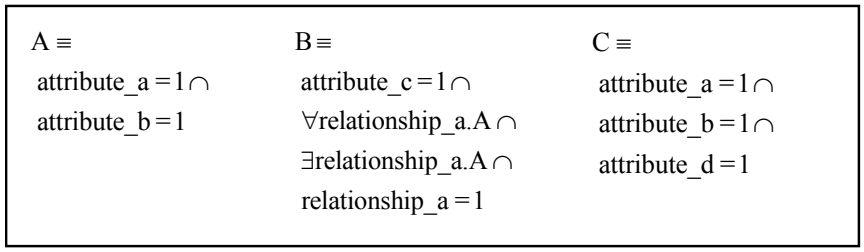


Figure 3-5: Description logic concept definitions

The value of DL formalisms for conceptual modeling are the standard set of reasoning algorithms within the knowledge base at the intensional level (TBox) and the assertional level (ABox) [26]. Reasoning involves the identification of implicit knowledge based on what is explicitly stated in the model. The reasoning algorithms in DL include the following:

Schema consistency is checked by ensuring a nonempty database satisfies all constraints in the schema. Checking schema consistency is quite straightforward in simple semantic data models, but becomes more difficult with complex data models [33]. As previously stated, engineering information management systems tend towards complex schema. Schema consistency is based on the consistency of all concept definitions in the knowledge bases.

Concept consistency (concept satisfiability) is determined by checking if a concept cannot have an individual because it is over-constrained. Concept consistency helps during data design to ensure the schema is correct and can guide the designer to relax or change constraints. Concept consistency is a specialization of concept subsumption.

Concept equivalence determines if two classes denote the same set of individuals in the knowledge base. Equivalent concepts can be merged and the complexity of the knowledge base can be reduced.

Concept subsumption determines if a concept is subsumed by another concept. Concept subsumption addresses classification of concepts into a hierarchical taxonomy. Concept subsumption established the subclass-superclass relationships between concepts based on concept definitions.

3.6.4 Critical Analysis of Description Logic for Information Modeling in Engineering Design

Previously, the characteristics of engineering design problems and information modeling requirements are identified. In this section we critically evaluate DL for information modeling in the context of the key EIM requirements. A summary of information modeling formalisms is presented in Table 3-4.

Table 3-4: Summary of modeling formalisms

	EIM Requirement		
Formalism	**R1:** Extensible	**R2:** Consistent	**R3:** Organization
SDM	• Not easy to extend schema – focus on static data representations	• Manual effort required for checking schema	• Does not support hierarchical organization
OODM	• Difficult to extend classes	• Consistency not guaranteed	• Explicit subclass definition
DL	• Vocabulary and constructs enable extensibility • Easily modified	• Consistency between concepts is checked	• Classification and organization of concepts

In Table 3-5, a correlation between information modeling formalisms and EIM requirements is summarized, followed by a detailed discussion.

Table 3-5: Summary of modeling formalisms and EIM requirements

	EIM Requirement		
Formalism	**R1:** Extensible	**R2:** Consistent	**R3:** Organization
SDM	○	○	○
OODM	●	◎	○
DL	●	●	●

Extensibility (R1): It is shown that new concepts can be defined in the information model at anytime and in any order. As new concepts are added or existing concepts are modified the DL reasoning algorithms will reclassify the concepts and create a new hierarchy based on the concept definitions. Unlike SDM and OODM which require subclass relationships to be explicitly stated, DL utilizes reasoning services for organizing the information hierarchically independent of concept definition. SDM does not support the notion of extensibility of the information schema, rather an information model can only be used after is it completely specified. In DL, the information model can be extended and automatically reclassified. While OODM enables hierarchical taxonomies to be created through *typing,* it does not fully support the notion of extensibility.

Information consistency (R2): Information consistency in engineering information models is important to ensure that the concepts defined by multiple designers in the information model are correct. Information consistency is closely related to R1 and R7. In this context, we are primarily concerned with the internal consistency of the concept definitions (i.e., TBox level). As discussed in Section 3.3, standard DL reasoning algorithms can be leveraged to check the consistency of concept definitions as they are

defined. With semantic and object-oriented modeling, checking the internal consistency of the schema is largely an *ad hoc,* manual effort. As the number of concepts and properties in the information model increases, human error can play an important role in correctly identifying inconsistent concepts. Thus, automatically detecting information consistency using mathematical algorithms is both quicker and provides a more complete check. This is especially useful in multi-disciplinary decision making when different designers are creating decision concepts using a shared vocabulary.

Provide information organization capabilities (R3): DL reasoning algorithms are used to organize the concepts specified in the information model in hierarchical taxonomy. Subsumption, the most basic reasoning algorithm is used to determine the implicit relationships between concepts definitions. This simple, yet powerful ability enables designers to describe the properties of concepts, not focus on the structure, organization, and relationship with other concepts. DL reasoners are used to determine the implicit subclass/superclass relationships between concepts. In SDM and OODM subclass/superclass relationships must be explicitly stated in the schema. Information modelers must ensure the relationships are explicitly stated for creating hierarchies. Additionally, the order in which concepts are created plays an important role in the overall structure and ease of creating hierarchical taxonomies. For complex information models, designers may not explicitly state these relationships and affect the richness of the representations.

The goal of this research is not to extend or augment DL, but rather (1) critically assess the value of DL for engineering information management, and (2) utilize DL for

developing information models of engineering design decisions. A detailed discussion of Description Logic is presented in [16].

3.6.5 Description Logic Development Technologies

In this section the technologies for developing DL information models are presented. An introductory explanation of the technologies used in this work is included in this section. Four components make up the *development* environment for the decision-centric design knowledge representation framework. The architecture and interface of the components is illustrated in Figure 3-6.

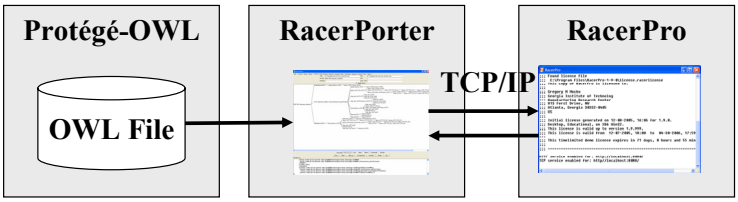


Figure 3-6: Architecture and interface between components in the knowledge base

Protégé-OWL – Protégé-OWL editor is an extension of Protégé that supports developing ontologies using the Web Ontology Language (OWL). Protégé-OWL is an open-source ontology development environment with functionality for editing OWL based ontologies. Protégé-OWL enables (1) create OWL ontologies, (2) visualize classes and properties, (3) define class definitions based on OWL expression, and (4) call DL reasoners.

RacerPro **–** RacerPro is a knowledge representation system that implements a highly optimized tableau calculus for very expressive DL. RacerPro provides algorithms for reasoning at the TBox and ABox level. RacerPro is the back-end reasoner used within Protégé-OWL. RacerPro implements the HTTP-based quasi-standard DIG for connecting

with Protégé-OWL. RacerPro is a knowledge representation system for DL. RacerPro implements an optimized tableau calculus algorithm for highly expressive DL languages. While RacerPro can be used for developing DL knowledge bases, it is primarily used in this research for reasoning services at both the TBox (concept terminology) and ABox (concept individual) level. RacerPro supports the $\mathcal{SHIQ}$ (equivalent to $\mathcal{ALCQHI}_{\mathcal{R}+}$) representation, although less expressive languages can be used. The $\mathcal{SHIQ}$ language extends the basic $\mathcal{ALC}$ language through the inclusion of additional restrictions and axioms including qualifying number restrictions, role hierarchies, inverse relationships, and transitive roles. Similar to other knowledge-based systems, RacerPro is based on the open world assumption (OWA). The OWA states that *what is not explicitly stated in the knowledge base cannot be proven to be true or false.*

RacerPorter *(see Figure 3-7)* – RacerPro is usually used as a back-end reasoner for other applications and is accessed through the DIG interface. However, it was determined that the Protégé-DIG-RacerPro interface was limited. Hence, RacerPorter is used as the graphical interface to RacerPro. RacerPorter connects to RacerPro through a TCP/IP interface. RacerPorter Enables OWL ontologies to be loaded and reasoned with at the T-Box and A-Box level.

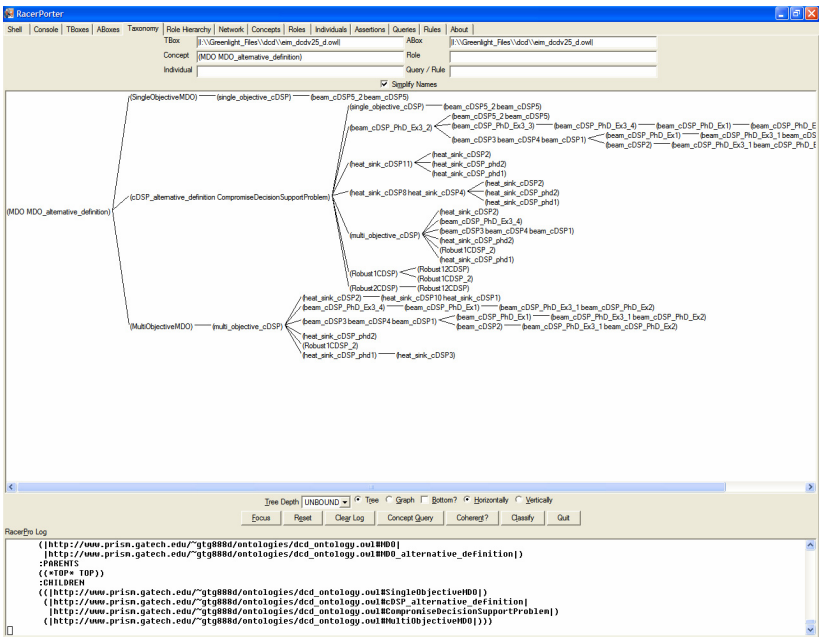


Figure 3-7: RacerPorter interface

OWL-DL File Storage – OWL-DL is a standard XML-based language that is used for explicitly representing the meaning of terms in vocabularies and the relationships between those terms. OWL DL provides support for developing ontologies using DL representations. OWL is a standard ontology language by W3C. OWL is the markup language used for storing DL ontologies. The Web Ontology Language (OWL) is a representation mechanism designed at increasing the information content shared between agents, including computer applications and humans. OWL facilitates the representation of machine-processible knowledge. According to [1], OWL is intended to be used when information in documents must be processed by computer-based applications in addition to humans. OWL is developed based on several layers of "standardized" components in accordance with W3C (see Figure 3-8).

- XML provides a mechanism for representing the syntax of structured documents. XML does not capture any semantic constraints (meaning) of the document.

- XML Schema is a language for restricting the structure of XML documents and also extends XML with data types.
- RDF is a data model for objects ("resources") and relations between them, provides a simple semantics for this data model, and these data models can be represented in an XML syntax.
- RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes.
- OWL adds vocabulary for describing properties and classes that include relations between classes, cardinality, equality, and characteristics of properties.

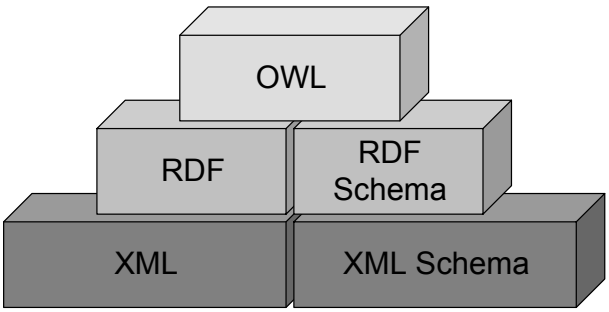


Figure 3-8: Web Ontology Language (OWL) Architecture

A detailed discussion and explanation of the OWL language is available at [1].

3.7 Discussion of cDSP Formal Language

Engineering information management is essential in the design and realization of modern complex systems. As such, EIM systems are needed for capturing, representing, and sharing information throughout the design process. A multitude of information models have been proposed for capturing a broad scope of product information ranging

from product geometry to configuration control. To realize an information model for capturing product-related design information, developers must commit to a particular information modeling formalism. The formalism chosen does not impose restrictions on the domain that is modeled, but does impose characteristics and limitations of the formalisms themselves. The resulting information representations are in a large part influenced and even limited by the underlying formalisms. Hence, particular attention should be given to the domain-independent modeling formalism prior to developing domain-specific information representations. The assessment and subsequent selection of a particular information modeling formalism should not be brushed over. Unfortunately, this is not often the case with the development of information models in general, but more specifically engineering information models.

Thus, in this chapter three commonly-accepted information modeling formalism are critically assessed in the context of developing engineering information models. First, several characteristics of engineering design problems are identified. From these characteristics a requirements list for engineering information management is developed. The requirements are used as the basis for evaluating the characteristics, strengths, and limitations of the information modeling formalisms. It is then argued that DL provides several advantages over other information modeling formalisms including information extensibility, consistency, and organization. DL provides a structured, logics-based formalism for describing information. This not only enables complex concepts to be specified within a domain, but also enables the concepts to be reasoned. The ability to reason enables engineering information modelers to ensure consistency during development and enables users to organize and retrieve information.

3.8 Verification and Validation

In this chapter, the theoretical structural validation (TSV) is discussed (see Table 3-6). TSV refers to accepting the validity of individual constructs used for explicitly representing engineering decision information. In this chapter, one foundational construct is presented - description logic (DL) as a formalism for formally representing information. The internal consistency of DL is checked by critically evaluating current research and development efforts and applications of DL for information modeling. However, in addition to gaining confidence in the applicability of DL for EIM, description logic provides a mathematical foundation for representing knowledge. Thus, we can take advantage of the mathematical properties in arguing TSV. For example, the mathematical properties of DL ensure sound and complete reasoning.

In Section 3.6.4, it is argued that DL is appropriate for modeling information associated with engineering design and offers advantages over other modeling formalisms. Based on existing literature, it is shown that DL has been used for conceptual information modeling in several domains including medical, configuration management, and database development [16]. However, from existing literature, it is identified that DL has not been widely used in engineering information management. While there is growing interest in DL for EIM, current researchers have failed to adequately establish the motivation for using DL over other representational formalisms. Thus, several characteristics of engineering design problems are identified for engineering information management. These characteristics are used to highlight the requirements for engineering design problems. From the set of requirements three different information modeling

formalisms are assessed. It is determined that DL provides several advantages over other modeling formalism for addressing engineering information management problem.

Table 3-6: Validation and verification in Chapter 3

Theoretical Structural Validation
§3.4 – Identification of characteristics for information modeling in engineering design. The characteristics of engineering design problems are identified for developing a requirements list.
§3.5 – Creation of requirements list for engineering information modeling. The requirements list provides a basis for assessing information modeling formalism for addressing engineering information management issues
§3.6 – Presentation and discussion of information modeling formalisms. The characteristics of the modeling formalisms are presented and discussed in the context of the characteristics and requirements identified in §3.4 and §3.5.
§3.6.4 – §3.6.5 - The internal consistency of DL for engineering information modeling is established. The mathematical properties of DL provide advantages for developing information models, including reasoning services. A primary goal for verification and validation in this chapter is establishing the internal consistency of DL for engineering information modeling.

The mathematical properties and performance of DL enable us to draw conclusions about the internal consistency of knowledge bases represented using description languages. Baader and colleagues [16] discuss the internal consistency and complexity of DL in depth. In this dissertation, the mathematical properties of DL provide support for internal consistency because of the completeness and soundness of reasoning algorithms. As previously shown in Table 3-3 and discussed in 3.6.3, the language used in this

research is ALCON which has a PSpace-complete complexity. The ALCON provide consistent and complete reasoning for satisfiability and subsumption at both the Tbox and Abox levels. Due to this logical procedure of the critical literature review, gap analysis, and assessment of the information modeling formalism, the theoretical structural validity of the DL is accepted.

3.9 Chapter Synopsis

In this chapter, information modeling formalisms are assessed in the context of engineering information management (see Figure 3-9). The information modeling formalisms are assessed based on several design problem characteristics and information management requirements. The aims proposed at the beginning of this chapter are addressed:

- ✓ To establish the motivation for developing a formalism for design decisions- The motivation for developing a formalism is established by critically evaluating the current state of information modeling in engineering design. Additionally, the characteristics of engineering design problems are discussed in the context of information management.
- ✓ To introduce the information modeling requirements for modeling design decisions. Information modeling requirements are developed based on the characteristics of engineering design problems. Several high-level requirements are established for evaluating modeling formalisms.

- ✓ To critically assess modeling formalisms for capturing information – A review of three information modeling formalisms is completed. The modeling formalism are qualitatively evaluated against the modeling requirements.
- ✓ To evaluate DL for addressing engineering design problems - Based on the evaluation DL is discussed in detail against the engineering design characteristics and requirements.

In this chapter, it is established that DL is a valuable representational formalism for engineering design information. In Chapter 4, an information model is developed using DL for capturing decision-related design information.

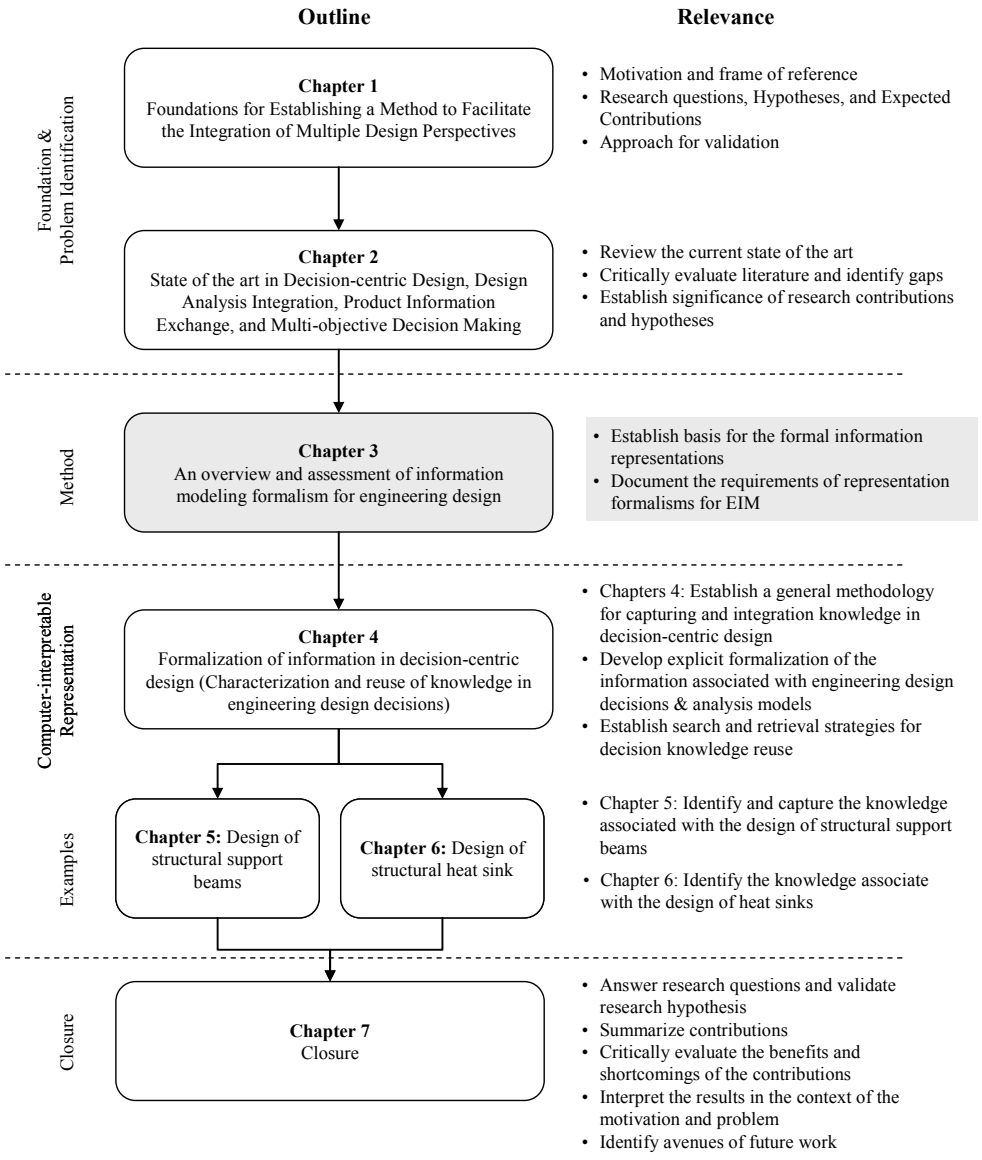


Figure 3-9: Outline of dissertation

CHAPTER 4:

COMPROMISE DECISION SUPPORT PROBLEM INFORMATION MODEL

Aims

- To introduce the information modeling requirements for modeling design decisions.
- To introduce a systematic method for formulating engineering design decisions.
- To introduce the basic vocabulary (the syntax and semantics) for describing compromise decision support problems.
- To illustrate how DL is used to specify several concept definitions of engineering design decisions.
- To critically assess the application of DL modeling in engineering design.

In this chapter a formal language for describing engineering design decisions is presented. A requirements list is developed to define the domain and scope of the information model. A method is then proposed as a systematic approach for formulating engineering design decisions. From the method, the information concepts and relationships associated with engineering design decisions are identified. An information model is developed based on the previously identified concepts and relationships. Finally, the DL representation of the compromise decision support problem is discussed. Several examples are presented in this chapter to illustrate the usage, robustness, and extensibility of the language (see Table 4-1).

Table 4-1: Summary of examples presented in this chapter

Example Concept	Description
• Compromise decision support problem (cDSP)	The vocabulary is developed for capturing the semantics of ht cDSP. The vocabulary is extracted from informal representations of the cDSP. Hence, the vocabulary is verified and validated by creating formal definitions of the cDSP concepts. The cDSP concept serves as the datum for discussion the vocabulary.
• Alternative definition cDSP	The cDSP is defined using the same base vocabulary, but with different complex concepts. An alternative definition is created to illustrate semantic equivalence and detection using DL reasoning.
• Single objective cDSP • Multi objective cDSP	The single objective and multi-objective cDSP concepts are specializations of the cDSP. The concepts are specified to illustrate the dynamic organization of the information base. Additionally, the concepts illustrate the consistency of the information base.
• Robust I cDSP • Robust II cDSP • Robust I and II cDSP	The extensibility and robustness of DL is illustrated by adding new terminology to the vocabulary to describe Robust design decisions. The modifications to the vocabulary are not propagated throughout existing concepts. However the information base is updated based on new concept definitions.
• AnalysisModel • EquationBasedAnalysisModel • ComputationalAnalysisModel	The declarative information associated with engineering analysis models is represented using the vocabulary. The vocabulary is demonstrated to represent general engineering analysis models and specializations including equation-based and computational analysis model.

In the following sections, the objective is to develop a language for describing engineering design decisions, specifically the compromise decision support problem and

close variants. The examples summarized in Table 4-1 are discussed in appropriate sections.

4.1 Information Model Requirements of Engineering Design Decisions

In Chapter 3, several requirements were identified at the meta-level information formalisms. As a result of the information modeling assessment in Chapter 3, it was determined that DL fulfills the requirements of problems encountered in engineering information management. However, the requirements identified in Chapter 3 do not establish the scope of the information model and address the particular information modeling needs. Thus, in this chapter we are primarily concerned with identifying the requirement that define the purpose and scope of the information representation. In the context of the ontology development framework, originally presented in Chapter 3, the intended uses, the domain, and the purpose of the information model are established. From this domain and scope, the information modeling concepts are clearly identified (see Figure 4-1).

Three requirements identified in Chapter 3 pertain to assessing and identifying the underlying information modeling formalisms for engineering design. While the requirements are important in developing information models for engineering design problems, they are not specific for engineering decision information models. Hence, several requirements are identified for engineering design decisions. The requirements are developed for explicitly formulating the intended use and domain of the information model. The requirements provide the basis for assessing the "quality" of the information models. In addition to the generic criteria proposed by Gruber [60], the requirements provide several specific criteria for judging the "usefulness" of the information and

verifying and validating the research contributions. As illustrated in Figure 4-1, developing a set of requirements.

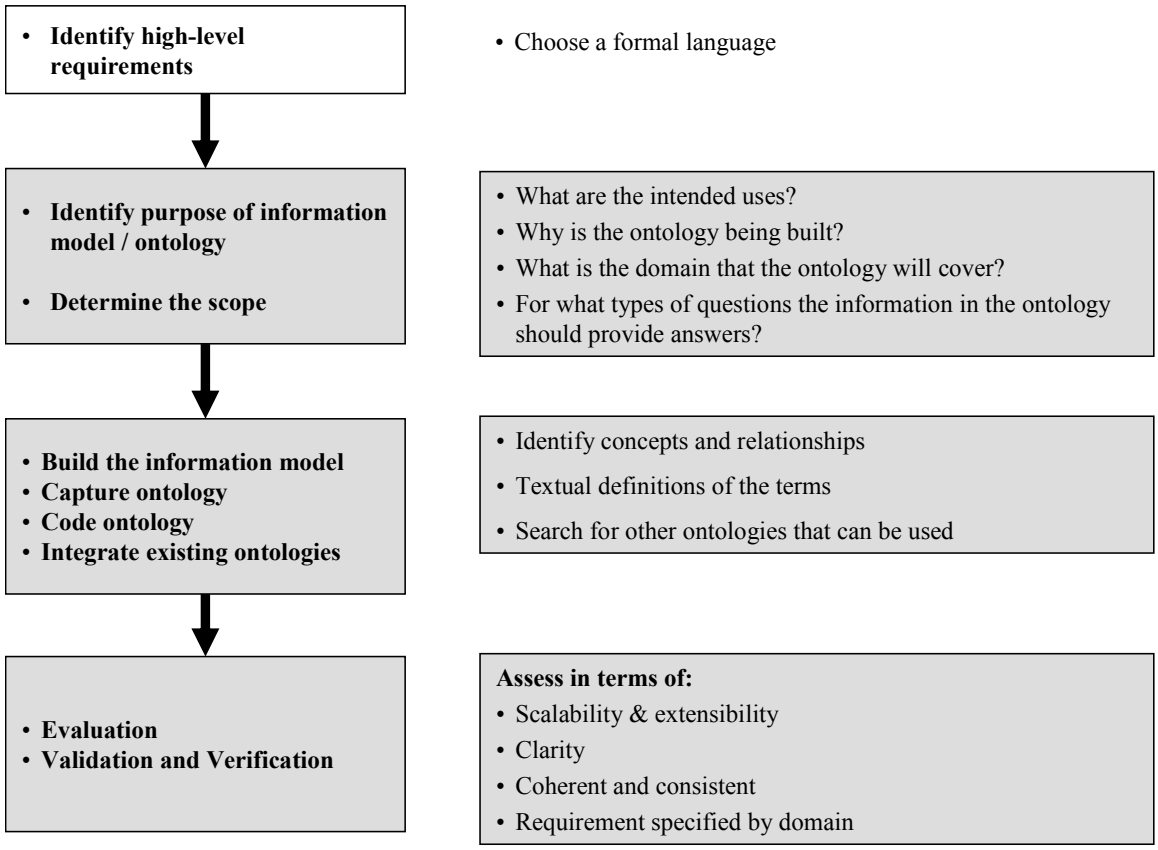


Figure 4-1: Information model and ontology development process

The following requirements defined the scope and purpose of the information model.

- **R4. The information model should enable designers to systematically capture and represent design problem knowledge from multiple perspectives.** Engineering design decisions typically involve multiple stakeholders, each with their own objectives, goals, constraints, limitations, and underlying knowledge in a particular area. For example, a structural designer may be an expert and have deep knowledge in the domain of structural mechanics, whereas an aerodynamics designer may be an expert and have a deep understanding of fluid flow. Ultimately, the design

of an airplane wing is dependent on the interaction and decisions based on each of these domains. Consequently, the final design of a wing must be determined based on trade-offs between structural considerations and fluid dynamics. However, the structural designer is not an expert in fluid mechanics and vice-versa. It is clear from this scenario, that knowledge must be unambiguously exchanged in the context of design decisions from several perspectives. Developing an explicit representation of decision-related knowledge will enable decisions to be composed of integrated expertise. Furthermore, by explicitly representing the knowledge contained in models from multiple perspectives, a decision maker reduces the possibility of invalid decisions.

- **R5. The information model should provide a computer-interpretable means for representing decision-related knowledge that can be exchanged and shared amongst stakeholders.** The use of computers, extended networks, and information technology in engineering design is prolific. In the context of distributed, collaborative decision-making, computer models are used for simulating product behavior and representing product geometry, as well as solving and modeling multi-objective optimization problems. However, the formulation of the design decisions based on the integration of knowledge encoded in computer models from multiple perspectives is limited. To facilitate the integration of knowledge from multiple sources, a common language is required to describe concepts, attributes, and relationships in a domain of discourse. Knowledge representations have gradually moved from the realm of artificial intelligence to application domains – in this case the domain is decision-centric design. Description Logic ontologies provide the

computational means for representing knowledge in the domain of interest. Ontology development defines a common vocabulary a particular domain and includes machine-interpretable definitions of basic concepts, attributed and relations among them.

- **R6. The information model should support the integration of analysis models from multiple disciplines and unambiguous representation of analysis-related knowledge.** Predicting the behavior of a system is an essential component in engineering decision-making because design objectives are often directly tied to the system behavior. The models used within engineering design decisions are the primary means for communicating knowledge across perspective. However, often analysis knowledge is shared between stakeholders in an ambiguous manner, often represented as black box relations. In this research, the explicit representation of analysis knowledge to support the seamless formulation of engineering design decision is essential.

- **R7. The information model should support the reuse and retrieval of decision-related knowledge.** In design, complex simulation models may be used to simulate the behavior of multiple products. For example, an Euler-Bernoulli beam model may be used to simulate the deflection of a crane under loading or the deflection of a floor joist in a residential building. In any event, complex simulation models are often reused across different product spaces. Similarly, existing design decisions may be used a basis for developing new decisions for similar products. For example, a heat exchanger may be optimized for thermal and structural considerations initially, and then additional phenomena, such as fatigue loading or buckling may be included as

goals of interest in a new design problem. To support reuse, two components are required (1) a schema for capturing decision-related knowledge and (2) reasoning services and algorithms to query and retrieve knowledge from the repository. To enable reuse of knowledge, design repositories have been proposed.

- **R8. The information model should have well-defined structure and pre-defined vocabulary of symbols to enable collaboration between decision makers.** The atomic concepts and concept properties should be unambiguously defined as well as the rules for putting the concepts together to define complex concepts. A well-defined vocabulary and grammar are essential to ensure that the knowledge exchanged between multiple disciplines is correct. An explicitly defined common vocabulary is essential for communicating and collaborating in design. The information should be represented unambiguously in a computational environment.

- **R9. The information model should enable the limitations and assumptions of analysis models to be captured.** The information model should support the representation of analysis model limitations. "Black-box" usage of analysis models is common in engineering design. However, not considering the assumptions and limitations of engineering analysis models can result in invalid design solutions. Thus, the information model must enable computational limitations, assumptions, and constraints on the analysis model to be captured and integrated with decision-related information.

- **R10. The information model should be easy to understand by engineering designer.** The engineering information models should be computer-processible and human interpretable. Engineering designers are experts in a particular field of design,

but not information modeling. Thus, in order to reduce the chance of misuse and miscommunication, the information models must be clear. The concepts and properties in the information model should be defined in a manner that is clear for engineering designers to use, thus enabling the use of the information model by designers with minimal expertise in information modeling.

4.2 Systematic Method for Formulating Engineering Design Decision

While the particulars for formulating a design decision often depend on the domain and type of decision being formulated, there is a need to capture the steps associated with problem formulation. A first step towards developing an information model of the cDSP is to explicitly model the phases and steps associated with formulating engineering design problems as cDSP. A systematic method for formulating multi-objective design problems in the form of cDSPs is proposed. The method facilitates representing decision related knowledge in a computational means from general knowledge about the design problem to structured knowledge of an engineering design decision. The method consists of seven phases (see Figure 4-2).

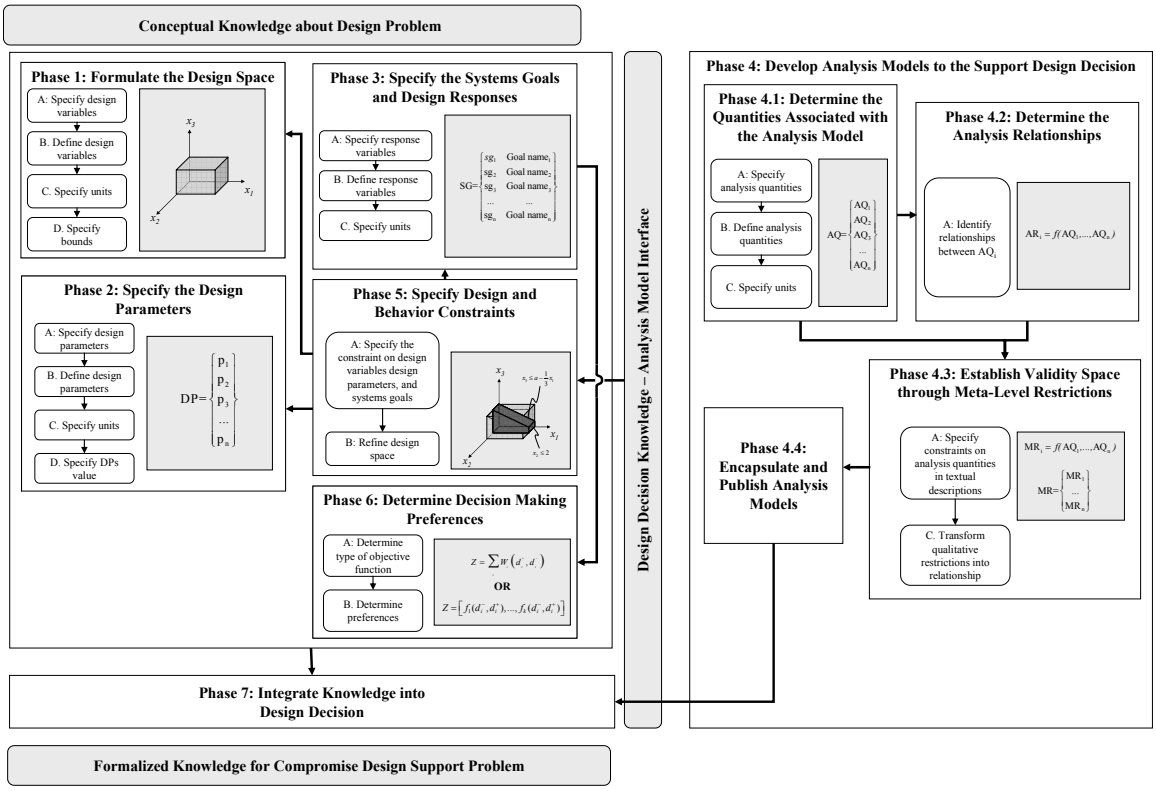


Figure 4-2: Method for formulating engineering design decisions

The method is decomposed into two sub-methods. The first sub-method is focused on the representation of decision-related knowledge for formulating design problems as compromise decision support problems and consists of Phases 1 – 3, 5, 6, and 7. The second sub-method is focused on the representation of engineering analysis models to support the focuses on the representation of analysis model knowledge to support design decisions. The analysis model characterization method consists of Phase 4. The method is decomposed in this manner because it is common in engineering decision making to use disciplinary code developed by domain experts. Several "mathematical" representations of the cDSP information are extracted based on the method. The mathematical representations are largely presented in matrix form.

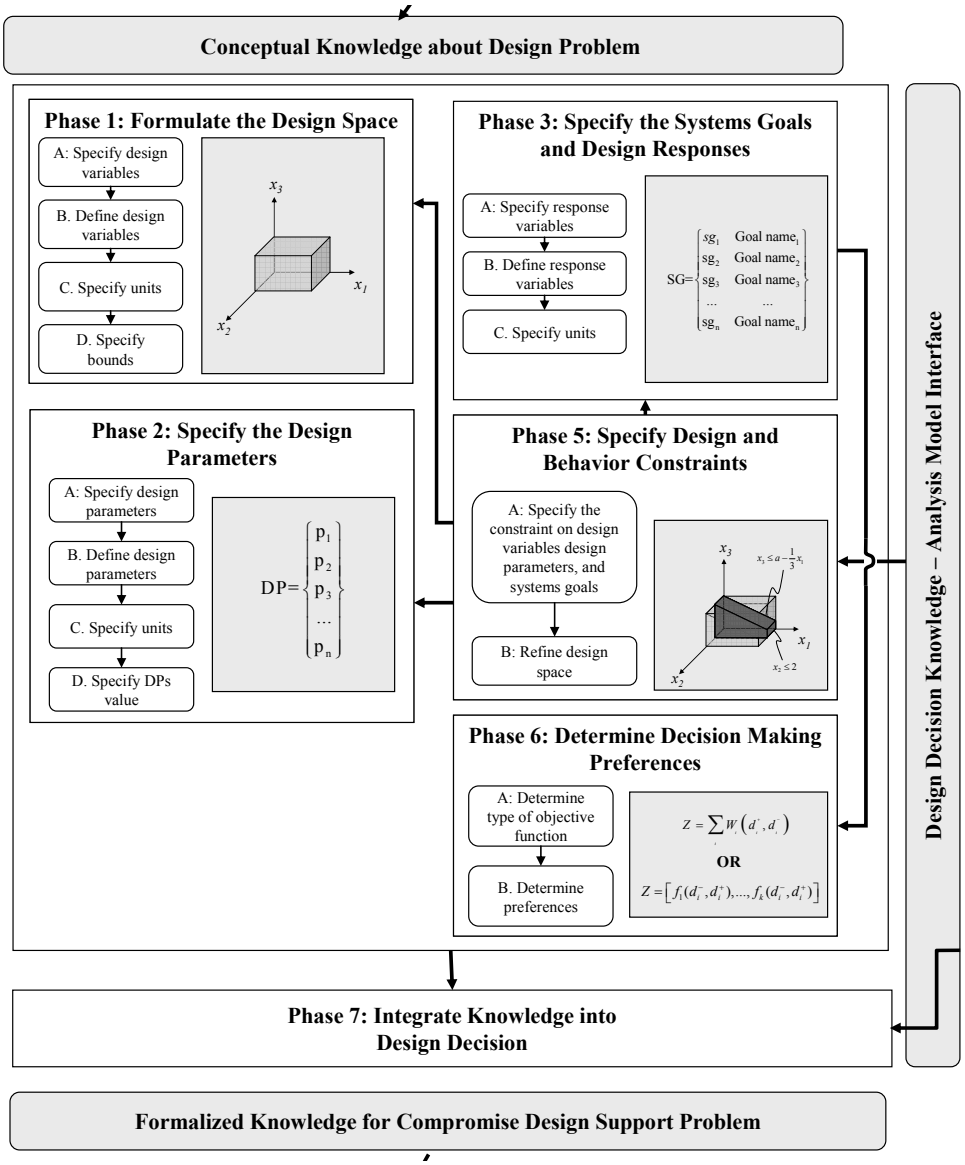


Figure 4-3: Systematic method for capturing decision related knowledge

Phase 1: Formulate the design space. First, the design space is formulated by specifying and a set of quantities that describe the system variables. The design space does not specify a particular solution to the design problem, but rather specifies the domain in which the solution exists. The design space must be completely and unambiguously defined. In other words, the design variables, definitions, units and bounds must be specified. The design space is limited to the variables of interest in the

design problem. The design space represents the domain in which a solution to the design problem may exist. The designer is responsible for specifying the design variables that describe the system of interest. The design space (DS) is specified by explicitly defining the system design variables of interest. The design space of interest for a system may change for different design problems. For example, the design variables associated with the structural rigidity of a structure may be different than the parameters associated with thermal performance. The coordination of design spaces between coupled decisions is not the focus of this research and thus not discussed in detail in this research. However, we realize that coupled decision making is a valuable research area and offer the following references of ongoing research efforts [76; 77; 80].The specification of the design spaces for a single design decision is completed in several steps, outlined below.

Step 1A: Specify the system design variables of interest. Identify the parameters of interest that define the product form in terms of system variables. The order of the design space is defined by the number of system variables that define the product. Designers define the design space by specifying what design variables are of interest to be modified. The designer does not specify what the values of the variable are, but rather describes at a conceptual level the system variables. The designer does not specify bounds on the design variables, but rather specifies what space is going to be explored in the design decision. Graphically, the design space is illustrated in Figure 4-4.

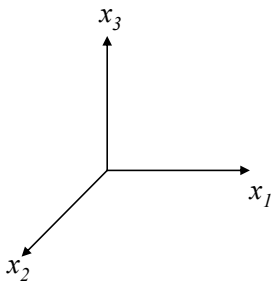


Figure 4-4: Initial specification of design space

As illustrated in Figure 4-4 the design space is defined as a space without restrictions or bounds. At this point, the designer wants to keep the design freedom as large as possible. The specified design variables define the space that can be explored in the design decision. The design variable space (DV) is defined by the set of design variables (dv) specified by the designers.

$$DV=\begin{Bmatrix} dv_1 \\ dv_2 \\ dv_3 \\ \dots \\ dv_n \end{Bmatrix} \qquad \textbf{4.1}$$

Step 1B: Define/select the descriptive names of the design variables. The names of the design variable are specified by the decision stakeholders. The names of the variables serve as unique identifiers of the variables that describe the system. The variable names may be selected from an established vocabulary for a particular domain of interest. For example, radius and diameter are used to describe the dimensions of a circle.

$$DV=\begin{Bmatrix} dv_1 & Variable\_name_1 \\ dv_2 & Variable\_name_2 \\ dv_3 & Variable\_name_3 \\ ... & ... \\ dv_n & Variable\_name_n \end{Bmatrix} \quad \textbf{4.2}$$

Step 1C: Define the units of the design variables. The design variable (DV) space must be completely defined in terms of system design variable. Thus, the design variables must be unambiguously specified. Thus, the units of the system variables must be specified in terms of a predefine set of units.

$$DV=\begin{Bmatrix} dv_1 & Variable\_name_1 & Unit_1 \\ dv_2 & Variable\_name_2 & Unit_2 \\ dv_3 & Variable\_name_3 & Unit_3 \\ ... & ... & ... \\ dv_n & Variable\_name_n & Unit_n \end{Bmatrix} \quad \textbf{4.3}$$

In this context, a unit is defined as a particular physical quantity, defined and adopted by convention, with which other particular quantities of the same kind are compared to express their value. The units of the design variables are specified from a predetermined ontology of units. For example, units may be specified from the Systems International (SI) system or British Engineering System. The units of the design variables must be explicitly defined to ensure proper representation of physical quantities. The units used to describe a system do not need to be from the same system of unit. However, conversion factors between the systems must be established.

Step 1D: Identify the bounds on the system parameters. The design space is refined through the specification of bounds on the design variables. The design space is previously specified at a high level. The high-level specification ensures

designer(s)/stakeholder(s) to not specify false restrictions on the design variables. Thus, the designer further refines the design space by specifying the bounds on the design variables. Bounds are typically numerical values that specify the upper and lower limits that the design variable can be. The bounds on the design variables are expressed mathematically as:

$$dv_i^{\min} \le dv_i \le dv_i^{\max} \qquad \textbf{4.4}$$

The bounds on the design variables are combined with the previously established representation (4.3), resulting in 4.5.

$$\text{DV}=\begin{Bmatrix} \text{dv}_1 & \text{Variable\_name}_1 & \text{Unit}_1 & dv_1^{\min} & dv_1^{\max} \\ \text{dv}_2 & \text{Variable\_name}_2 & \text{Unit}_2 & dv_2^{\min} & dv_2^{\max} \\ \text{dv}_3 & \text{Variable\_name}_3 & \text{Unit}_3 & dv_3^{\min} & dv_3^{\max} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \text{dv}_n & \text{Variable\_name}_n & \text{Unit}_n & dv_n^{\min} & dv_n^{\max} \end{Bmatrix} \qquad \textbf{4.5}$$

The bounded design space is illustrated graphically in Figure 4-5.

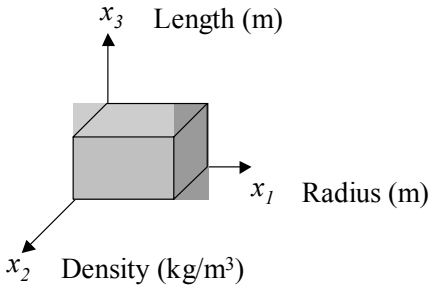


Figure 4-5: Refinement of design space by information about design variables

Phase 2: Specify the design parameters. In Phase 2, the design parameters associated with the design decision are established. Design parameters are distinguished

from design parameters to emphasize the quantities that define the *search* space within a decision and those quantities that are controlled external to the design (see Figure 4-6).

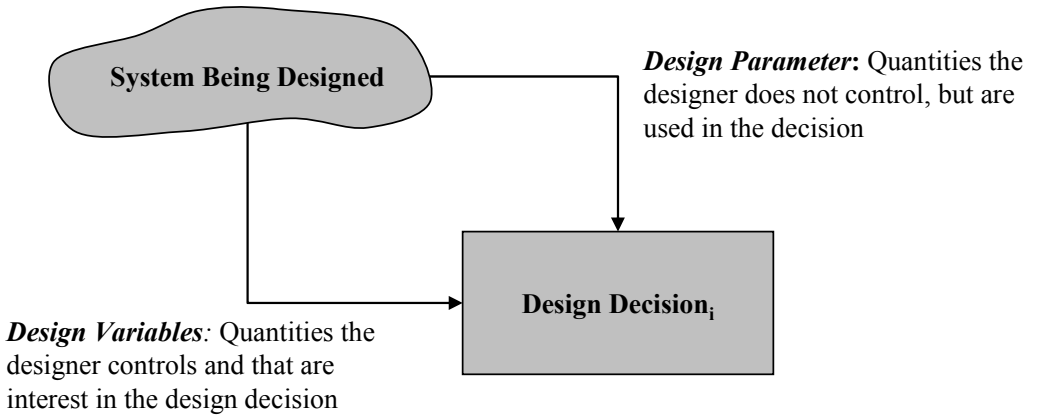


Figure 4-6: Distinguishing between design variables and design parameters in a design decision

Design parameters are similar to design variables, but the design parameters do no vary in the design decision. The design parameters are essential to the design decision, and must be fully defined in order to execute a design decision. Design parameters may describe the loading or boundary conditions, or characteristics of the systems that are not studied in a design decision. Design parameters may include constants (i.e., universal gravitational constant, pi, etc) or fixed quantities that describe the *form* of a system and its environment. Parameters must be defined, assigned values, and given units. It is important to note that parameters in one design decision may be specified as variables in other design decisions. The systematic approach for characterizing design parameters is based on the approach for characterizing design variables and the associated design space. The difference between the approaches is a space is defined for design variable whereas a single point is defined for design parameters. Design variables can vary over the design space during a decision and design parameters are assigned a single value for a

given decision. It is worth noting that a quantity that is a design parameter in one decision can become a design variable in another decision.

Step 2A: Specify the system parameters of interest. Identify the parameters of interest that define are associated with the system. Abstraction of essential design parameters are essential and must be correctly identified. The designer needn't specify *all* design parameters associated with the system, but only specify those design parameters that enable a design decision to be completed. The design parameters (DP) associated with the decision is defined by the set of design parameters (p) specified by the designers (see 4.6).

$$DP=\begin{Bmatrix} p_1 \\ p_2 \\ p_3 \\ \dots \\ p_n \end{Bmatrix} \qquad \textbf{4.6}$$

Step 2B: Define/select the descriptive names of the design parameters. The names of the design parameters are determined by the stakeholder in the design process. However, adhering to an established ontology will enable previous decisions to be queried and retrieved from a repository.

$$DP=\begin{Bmatrix} p_1 & Parameter\_Name_1 \\ p_2 & Parameter\_Name_2 \\ p_3 & Parameter\_Name_3 \\ \dots & \dots \\ p_n & Paramter\_Name_n \end{Bmatrix} \qquad \textbf{4.7}$$

Step 2C: Define the units of the design parameters. The units of the design parameters must be completely specified adhering to an established system of units.

$$DP=\begin{Bmatrix} p_1 & Parameter\_Name_1 & Unit_1 \\ p_2 & Parameter\_Name_2 & Unit_2 \\ p_3 & Parameter\_Name_3 & Unit_3 \\ ... & ... & ... \\ p_n & Parameter\_Name_n & Unit_n \end{Bmatrix} \quad \textbf{4.8}$$

Step 2D: Specify the values of the design parameters. Because the design parameters do not vary over a given decision the values of those parameters must be specified. Unlike the system variables in which a space is specified, the design parameters take specified values. Examples of design parameters in the design of a structural member may include loading from external systems, modulus of elasticity. Whereas, design variables in the design decision may include beam height and beam width. Instances of design parameters associated with a design problem are represented mathematically in 4.9.

$$DP=\begin{Bmatrix} p_1 & Parameter_1 & Unit_1 & Value_1 \\ p_2 & Parameter_1 & Unit_2 & Value_1 \\ p_3 & Parameter_1 & Unit_3 & Value_1 \\ ... & ... & ... & ... \\ p_n & Parameter_1 & Unit_n & Value_1 \end{Bmatrix} \quad \textbf{4.9}$$

Phase 3: Specify the Systems Goals and Responses. In Phase 3, the design goals are specified. In most cases, the design goals represent the behavior of the systems that is of interest. The system behavior, in this context is how the systems responds or reacts according to a model that represents the system. The behavioral response may range from cost relationships to physics-based calculations. In this phase, the goals that are of

interest in a particular design problems are identified. In the compromise decision support problem (cDSP), there are multiple goals.

Step 3A: Determine the response variables and system goals of interest in the design decision. Identify the behavioral responses that are of interest in the design decision. The system goals (SG) are defined by specifying the system responses completely.

$$SG=\begin{Bmatrix} sg_1 \\ sg_2 \\ sg_3 \\ ... \\ sg_n \end{Bmatrix} \qquad \textbf{4.10}$$

Step 3B: Define the names of the response objective. The names of the design response are determined by the stakeholder in the design process. It is common for design response to be chosen from an predetermined set of engineering phenomena. The level of detail of the design goal depends on the expertise of the designer. For example, a designer may specify that stress is a goal, or may specify that normal stress in the x-direction is the goal. It is illustrated that the latter concept is a more specific concepts. However, if a new behavior is of interest a new concept in the ontology must be defined.

$$SG=\begin{Bmatrix} sg_1 & \text{Goal name}_1 \\ sg_2 & \text{Goal name}_2 \\ sg_3 & \text{Goal name}_3 \\ ... & ... \\ sg_n & \text{Goal name}_n \end{Bmatrix} \qquad \textbf{4.11}$$

Step 3C: Define the units of the response variables. The units of the responses are determined according to an established system of units.

$$SG=\begin{Bmatrix} sg_1 & \text{Goal name}_1 & \text{Unit}_1 \\ sg_2 & \text{Goal name}_2 & \text{Unit}_2 \\ sg_3 & \text{Goal name}_3 & \text{Unit}_3 \\ ... & ... & ... \\ sg_n & \text{Goal name}_n & \text{Unit}_n \end{Bmatrix} \quad \textbf{4.12}$$

Phase 4: Develop Analysis Models to the Support Design Decision. Phase 4 marks the beginning of the second sub-method. The second sub-method is focused on the capturing and representing the knowledge associated with engineering analysis models. As previously stated, the characterization of analysis model knowledge is presented as an integral, but separate method to emphasize the use and development of external models to support engineering design decisions (see Figure 4-7).

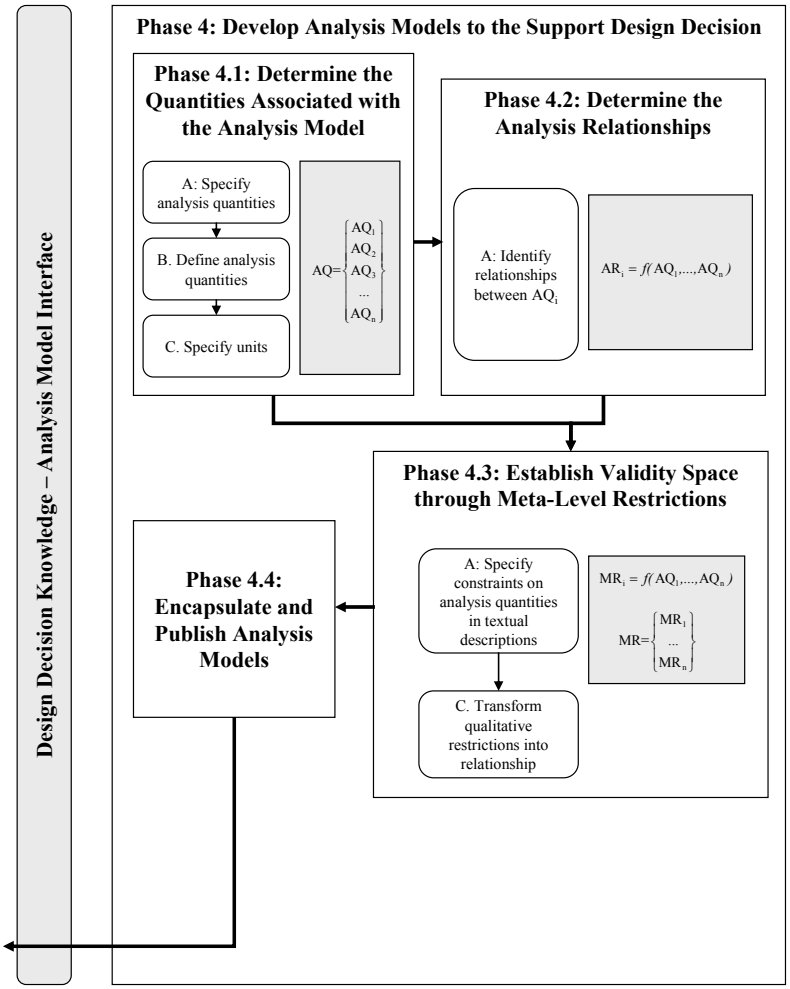


Figure 4-7: Systematic method for explicitly representing analysis model knowledge

Phase 4.1: Determine the Quantities Associated with the Analysis Model. The analysis quantity space (AQ) is specified by explicitly defining the quantity associated with an analysis models

Step 4.1A: Specify analysis quantities. Analysis quantities are those quantities that are required for executing the analysis model. In this context, the analysis models knowledge is assumed to be declarative, thus input and output quantities are not specified by the modelers. The analysis quantity space (AQ) is defined by the set of analysis quantities associated with a particular analysis model.

$$AQ=\begin{Bmatrix} AQ_1 \\ AQ_2 \\ AQ_3 \\ \dots \\ AQ_n \end{Bmatrix} \qquad \textbf{4.13}$$

Step 4.1B: Define the descriptive names of the analysis quantities. The names of the analysis quantities may be defined or selected from the information model. The analysis quantities may be selected from an established vocabulary for a particular domain of interest and will most likely be common with design variables and design parameters.

$$AQ=\begin{Bmatrix} AQ_1 & Quantity\_name_1 \\ AQ_2 & Quantity\_name_2 \\ AQ_3 & Quantity\_name_3 \\ \dots & \dots \\ AQ_n & Quantity\_name_n \end{Bmatrix} \qquad \textbf{4.14}$$

Step 4.1C: Define the units of the design variables. The analysis quantity space must be completely defined, thus, the units of the analysis quantities must be specified.

$$AQ=\begin{Bmatrix} AQ_1 & Quantity\_name_1 & Unit_1 \\ AQ_2 & Quantity\_name_2 & Unit_2 \\ AQ_3 & Quantity\_name_3 & Unit_3 \\ \dots & \dots & \dots \\ AQ_n & Quantity\_name_n & Unit_n \end{Bmatrix} \qquad \textbf{4.15}$$

Phase 4.2: Identify and Determine the Analysis Relationships. The analysis relationships (AR) capture the mappings between analysis quantities. The relationships

are specified at a sufficient high level to account for many different types of analysis relationships and to account for equations-based representation or complex analysis code. For example, pointers to external executable code may be included in the analysis relationship specification. The goal of specifying the analysis relationships is not to replace existing and legacy analysis code, but rather encapsulate the code in reusable format. An analysis model comprises a single analysis relationship.

$$AR_i = f(AQ_1, ..., AQ_n) \qquad \textbf{4.16}$$

Phase 4.3: Establish Validity Space through Meta-Level Restrictions. The meta-level restrictions (MR) represent the implicit assumptions and computational limitations of the analysis models and establish the validity space over which the analysis model can be used with confidence. Several MRs may be associated with a single analysis relationship.

Step 4.3A: Specify constraints on analysis quantities in textual descriptions. The meta-level relationships are first described in lexical format. A textual description of the analysis constraints and limitations is the first step in capturing the limitations of analyses models. The textual description describe the meta-relationships in terms of analysis quantities (AQ), design parameters (DP), and system design variables (DV)

$$MR_i = \text{Textual Description} \qquad \textbf{4.17}$$

Step 4.3B: Transform qualitative restrictions into constraints. Analytical relationships are realized based on the textual description of the meta-level constraints.

$$MR_i = \{\text{Textual Description} \quad f(AQ_1, ..., AQ_n)\} \qquad \textbf{4.18}$$

Several meta-level constraints can be associated to a particular analysis model. The meta-constraint consists of the entire set of assumptions and limitations

$$MR = \begin{Bmatrix} MR_1 \\ ... \\ MR_n \end{Bmatrix} \qquad \textbf{4.19}$$

Phase 4.4: Encapsulate and Publish Analysis Models. The knowledge associated with the analysis models (AM) include the analysis relationship and the meta-level constraints. The knowledge is published to an information model to facilitate integration and reuse in engineering decisions.

$$AM = \{AR \quad MR\} \qquad \textbf{4.20}$$

Phase 5: Specify Design and Behavior Constraints: A system constraint models the limits placed on the design. The set of constraint must be satisfied for the design feasibility. System constraints are specialized into two different types. A behavioral requirement constraint captures the relationships between the response space and design parameters. For example, the maximum deflection at the free end of a cantilever beam

may be specified as: $\delta \le \delta_{Max}$. In this case, δ_{Max} is a system parameter and δ is a response goal. Behavioral requirement (BR) constraints do not contain relationships between system variables and system response; these are classified as analysis relationships.

The second type of constraint is the design requirement constraint. Design requirement constraints are similar to bounds, and may be used interchangeably. Design requirement (DR) constraints capture the relationships between design parameters and design variables. For example, the maximum width of the beam must be less than a specified maximum and is represented as: $b \le b_{\max}$.

Step 5A: Specify constraint on design parameters, design variables, and system goals. The behavioral model and design requirement constraint are represented in the same manner as the analysis relationships.

$$\mathrm{BR_i} = \text{Textual Description} \qquad \textbf{4.21}$$

$$\mathrm{DR_i} = \text{Textual Description} \qquad \textbf{4.22}$$

Step 5B: Transform qualitative restrictions into constraints. Analytical relationships are realized based on the textual description of the meta-level constraints.

$$\mathrm{BR_i} = \{\text{Textual Description} \quad f(\text{SG and DP})\} \qquad \textbf{4.23}$$

$$\mathrm{DR_i} = \{\text{Textual Description} \quad f(\text{DV and DP})\} \qquad \textbf{4.24}$$

Phase 6: Determine Decision Making Preferences. The objective in the cDSP is to minimize the deviation function. The deviation function in the cDSP is a function of the deviation variables. Deviation variables are computed for each of the systems goals based on the actual system goal and the target goal. A designer must set the aspiration level for the system goal in terms of target system values. A target goal value is specified for each of the system goals.

Step 6A: Determine type of objective function: The designer must choose the type of deviation function formulation as either Archimedean or Preemptive. Archimedean is based on weighting of each of the system goals, whereas Preemptive is based on rank ordering.

$$\text{DF} = \{\text{Archimedean} \cup \text{Preemptive}\} \qquad \textbf{4.25}$$

where DF is the deviation function. The deviation function can either take the Archimedean or the Preemptive formulation. The Archimedean formulation is commonly represented as a linear weighting function of the deviation variables.

$$Z = \sum_{i=1}^{m} \left(w_i^+ d_i^+ + w_i^- d_i^- \right) \qquad \textbf{4.26}$$

The preemptive formulation is a rank ordering of the deviation variables for the systems goals and is given as:

$$Z = \left[f_1\left(d_i^+, d_i^-\right), \ldots f_k\left(d_i^+, d_i^-\right) \right] \qquad \textbf{4.27}$$

Step 6B: Determine preferences. The formulations for the deviation function dictate the type of relative importance assigned to the system goals. The preferences are

determined by specifying the relative importance of the system goals and the target goal values. In the Archimedean formulation the relative weighting (w_i) of the system goals are specified, and in the Preemptive formulation the ranking (r_i) of the system goals are specified.

$$\text{Archimedean formulation} = \begin{Bmatrix} sg_1 & w_1 \\ sg_2 & w_2 \\ sg_3 & w_3 \\ \dots & \dots \\ sg_n & w_n \end{Bmatrix} \quad \textbf{4.28}$$

$$\text{Preemptive formulation} = \begin{Bmatrix} sg_1 & r_1 \\ sg_2 & r_2 \\ sg_3 & r_3 \\ \dots & \dots \\ sg_n & r_n \end{Bmatrix} \quad \textbf{4.29}$$

Finally, the target goal values are specified for each of the system goals:

$$SG = \begin{Bmatrix} sg_1 & \text{Goal name}_1 & \text{Unit}_1 & \text{Target Goal}_1 \\ sg_2 & \text{Goal name}_2 & \text{Unit}_2 & \text{Target Goal}_2 \\ sg_3 & \text{Goal name}_3 & \text{Unit}_3 & \text{Target Goal}_3 \\ \dots & \dots & \dots & \dots \\ sg_n & \text{Goal name}_n & \text{Unit}_n & \text{Target Goal}_n \end{Bmatrix} \quad \textbf{4.30}$$

Phase 7: Integrate Knowledge into Design Decision: The final phase of the decision problem formulation involves integrating the design information into a unified construct. The cDSP construct is given as:

$$\text{cDSP}=\{\text{DV, DP, SG, AM, DR, BR, DF}\} \qquad \textbf{4.31}$$

where DV are the design variables, DP are the design parameter, SG are the system goals, AM are the set of analysis models, DR are the design requirement constraints, BR are the behavioral requirement constraints, and DF is the deviation function.

In this following section the vocabulary for describing the `CDSP` and `AnalysisModel` concepts and properties are defined. The vocabulary is defined based on the mathematical concepts extracted from the systematic method.

4.3 Concepts and Properties for Describing Design Decision and Analysis Models

The concepts and concept definitions defined in the language are presented in Table 4-1. These concept definitions are the vocabulary for describing decision constructs. The concepts and properties presented in Table 4-1 and Table 4-2 are related to the mathematical terms defined in Section 4.2. The relationships between the concepts and properties and the mathematical term are denoted in parenthesis.

Table 4-2: Concept definitions for cDSP and Analysis Model

Concept	Concept Definition
AnalysisModel (AM)	A general concept that represents an engineering analysis model. Serves as a “interface wrapper.
CDSP (cDSP)	Class of engineering decision problems in which the deviation from a target goal is minimized while satisfying constraints and bounds.
ConstraintRelationship	Captures relationships between Quantity concepts.
Quantity	A quantifiable or assignable property ascribed to a particular phenomenon, body, or substance.
DecisionPreference	Captures the decision making preference. The DecisionPreference concept may be either Archimedean of Preemptive
Unit	A particular physical quantity, defined and adopted by convention, with which other particular quantities of the same kind are compared to express their value
Value	The numerical value of a Quantity
DeviationVariable	A measure of the deviation of the actual goal from the target goal
DeviationFunction (DF)	The DeviationFunction in the cDSP is a function of the deviation variables.
Relationship	Captures the actual relationship of a ConstraintRelationhsip

In addition to concepts, properties are used to create associations between concepts. As previously, stated properties are binary predicates that relate the concepts together

using an established semantics. The following properties are defined for representing engineering design decision and analysis models.

Table 4-3: Property definitions for cDSP and Analysis Model

Property	Property Definition
function_of	Specifies an association between a constraint relationship and a quantity. D: ConstraintRelationship, DeviationVariable, R: Quantity, Value
computationallimitation	Specifies the computation limitations of analysis models D: AnalysisModel, R: ConstraintRelationship
analysisrelationship	An analytical relationship is a representation of the behavior of the system. D: AnalysisModel, R: ConstraintRelationship
analysismetarelationship	An analysis meta relationship captures the assumptions of analysis models D: AnalysisModel, R: ConstraintRelationship
behavioralrequirement	A behavioral requirement is related to the behavior of the system and how the system must function. D: CDSP, R: designparameter.Quantity & systemgoal.Quantity
designrequirement	A design requirement is related to physical constraints imposed on the design. D: CDSP, R: designparameter.Quantity & designvariable.Quantity
relativeimportance	A relationship between a systemgoal and a DecisionPreference concept D: DeviationVariable, R: DecisionPreference
supportmodel	Links analysis models to engineering design decisions. D: CDSP, R: AnalysisModel

Table 4-2: Property definitions for cDSP and Analysis Model (continued)

designparameter (DP,p)	A system design parameter does not vary in the design decision, it may be a constant or a value set outside the scope of the decisions D: CDSP, R: Quantity
designvariable (DV, dv)	A system design variable is a quantity that is controllable by the designer. A system design variable can be continuous, discrete, or Boolean. D: CDSP, R: Quantity
systemgoal (SG)	A system goal is an objective of interest in the design decision. A system goal relates a decision model and a quantity D: CDSP, R: Quantity
targetgoal (SG)	A target system behavior value as set by the designer. D: Quantity, R: Value
has_unit (Unit)	Relates a quantity to unit concept D: Quantity, R: Unit
has_value	The value of a physical quantity is the quantitative expression of a particular physical quantity D: Quantity, R: Value
lowerbound (dv^{min})	A lower bound is a minimum lower value placed on system design variable. D: Quantity, R: float data type
upperbound (dv^{max})	An upper bound is a maximum upper value placed on system design variable. D: Quantity, R: float data type
relationshipequation	Describes the analysis relationship. D: ConstraintRelationship, R: Relationship
symbol	The symbol of a quantity D: Quantity, R: string data type

The vocabulary defined in Table 4-1 and Table 4-2 is used to create information models of the concepts associated with engineering design decisions. In the following section, the graphical representations of the information models are presented.

4.4 Information Modeling of Foundational Constructs

The vocabulary for representing engineering decision uses two basic concepts for describing more complex concepts. The `Quantity` concept is a foundational concept used in various specifying the `CDSP` and `AnalysisModel` concepts. The `Quantity` concept represents the physical quantities associated with engineering design and analysis models. The `Quantity` concept represents a physical quantity and is defined as:

A quantity in the particular sense is a quantifiable or assignable property ascribed to a particular phenomenon, body, or substance. Examples are the mass of the moon and the electric charge of the proton. A physical quantity is a quantity that can be used in the mathematical equations of science and technology [152].

The `Quantity` concept is shown graphically in Figure 4-8.

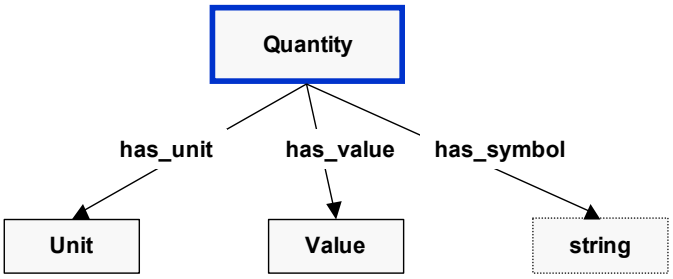


Figure 4-8: Graphical representation of Quantity concept

The Quantity concept is composed of has_unit, has_symbol, and value properties. The has_symbol property is a datatype property that takes a string value. The has_unit property is a object property that relate a Quantity to a Unit. Finally, the value property relates a Quantity to a Value concept. The Quantity concept enables engineering designers to unambiguously specify the quantities associated with design decisions and analysis models. The DL representation of the Quantity concept is given as:

```
Class(Quantity complete
and(
  ((∃ has_unit.Unit)∩(∀ has_unit.Unit)∩(= has_unit 1))
  ((∃ value.Value)∩(∀ value.Value)∩(= value 1))
  (= has_symbol 1)))
```

The description of a Quantity concept is described in prose as:

"A Quantity has at least one unit that is Unit and where all units are Units and has exactly one unit and has at least one value that is Value and where all values are Values and has exactly one Value and has exactly one symbol."

The ConstraintRelationship concept is used in developing definitions for the CDSP and AnalysisModel concepts. The ConstraintRelationship concept is a generic container for capturing the associations between quantities. The information model of the ConstraintRelationship concepts is illustrated in see Figure 4-9.

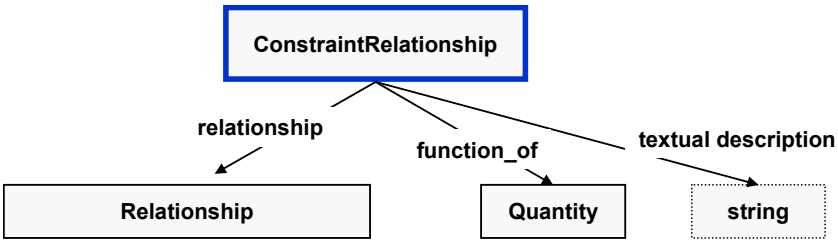
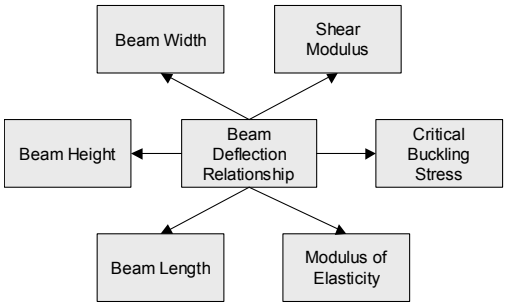
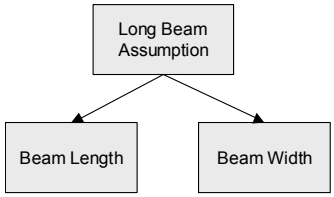
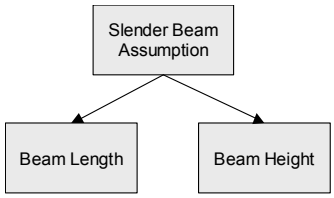
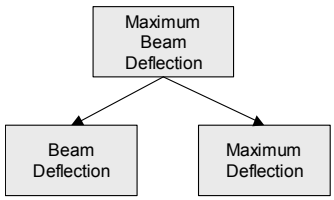


Figure 4-9: Graphical representation of ConstraintRelationship concept

The `ConstraintRelationship` relationship concept does not capture relationships as executable or procedural functions, but rather provide a generic declarative description of quantities associated with a constraint. In this context, the `ConstraintRelationship` concept enables a variety of constraint to be represented for simulating the behavior and capturing design or behavioral requirements.

The basic idea of a `ConstraintRelationship` is that quantities and relationships between those quantities can be represented in a declarative fashion, without specifying the causality of the relationships. In this research, the `ConstraintRelationship` concept is similar to constraint graphs and ideas presented by Peak [112]. The `ConstraintRelationship` concept enables "interfaces" between disciplinary analysis models to be specified for use in engineering decisions. Examples of the `ConstraintRelationship` concepts are presented in Table 4-4.

Table 4-4: Examples of ConstraintRelationship concepts

A. The deflection at the free end of a cantilever beam subjected to an end load - $\delta = \frac{4PL^3}{Ebh^3}$	 BeamDeflectionRelationship = $f(\delta, P, L, E, b, h)$
B. The Euler assumptions for the cantilever beam end load • Long beam assumption - $L \geq 10b$ • Slender beam assumption - $L \geq 10h$	 LongBeamAssumption = $f(L, b)$  SlenderBeamAssumption = $f(L, h)$
C. A design requirement stating the maximum deflection of the beam must be less than a specified value - $\delta \leq \delta_{Max}$	 MaximumBeamDeflection = $f(\delta, \delta_{Max})$
Note: All directed links represent `function_of` relationships	

As illustrated in Table 4-3, the `ConstraintRelationship` concept is used to capture analytical relationships between design quantities, meta-level relationships, and

design requirements. The DL representation of the ConstraintRelationship is given as:

```
Class(ConstraintRelationship complete
and(
  (∃ function_of.Quantity)
  (∀function_of.Quantity)
  (∃ relationship.Relationship)
  (∀ relationship.Relationship)
  (= relationship 1)
  (= textualdescription 1)))
```

The ConstraintRelationship concept is given in English as:

"A constraint relationship has exactly one relationship that is a Relationship, where all relationships are Relationships and exactly one textual description."

A detailed discussion of the limitations and implications of modeling the Quantity and ConstraintRelationship concepts is included in Section 4.8.

4.5 Information Modeling of Engineering Design Decisions

The concepts and properties are used in conjunction with DL constructs to define complex concepts for representing engineering design decisions. Several examples of decision constructs are presented to illustrate the extensibility and robustness of the language. Additionally, DL reasoning is utilized for organizing, checking consistency of the defined concepts.

4.5.1 Compromise Decision Support Problem (cDSP) Information Representation

The cDSP is a generic class of constrained multi-objective decision problems for modeling engineering decisions [91]. The keywords and descriptors of the cDSP are presented in Table 4-5.

Table 4-5: Keywords and descriptors for cDSP

Keywords	Descriptors
Given	System **design parameters** and constants **Analysis models** necessary for evaluating the goals, constraints and bounds, and the deviation function
Find	System **design variables** and deviation variables
Satisfy	**Systems goals**, **constraints** and **bounds**
Minimize	A deviation function

The keywords and descriptors and the structure presented in Table 4-5 serve the general framework for implementing the cDSP in DL. However, as illustrated in the following examples, the representation is changed from a largely flat structure to a layered structure. In other words, the relationships between the descriptors given in Table 4-5 are not accurately captured.

The graphical representation of the cDSP enables the information "network" in the design decision to be viewed. Additionally, the information model provides a foundation for implementing in DL. The graphical representation of the information model is similar to the ER diagram and represents the concepts (entities) and properties (relationship) between concepts. The graphical information model representation of the cDSP concept is presented in Figure 4-10.

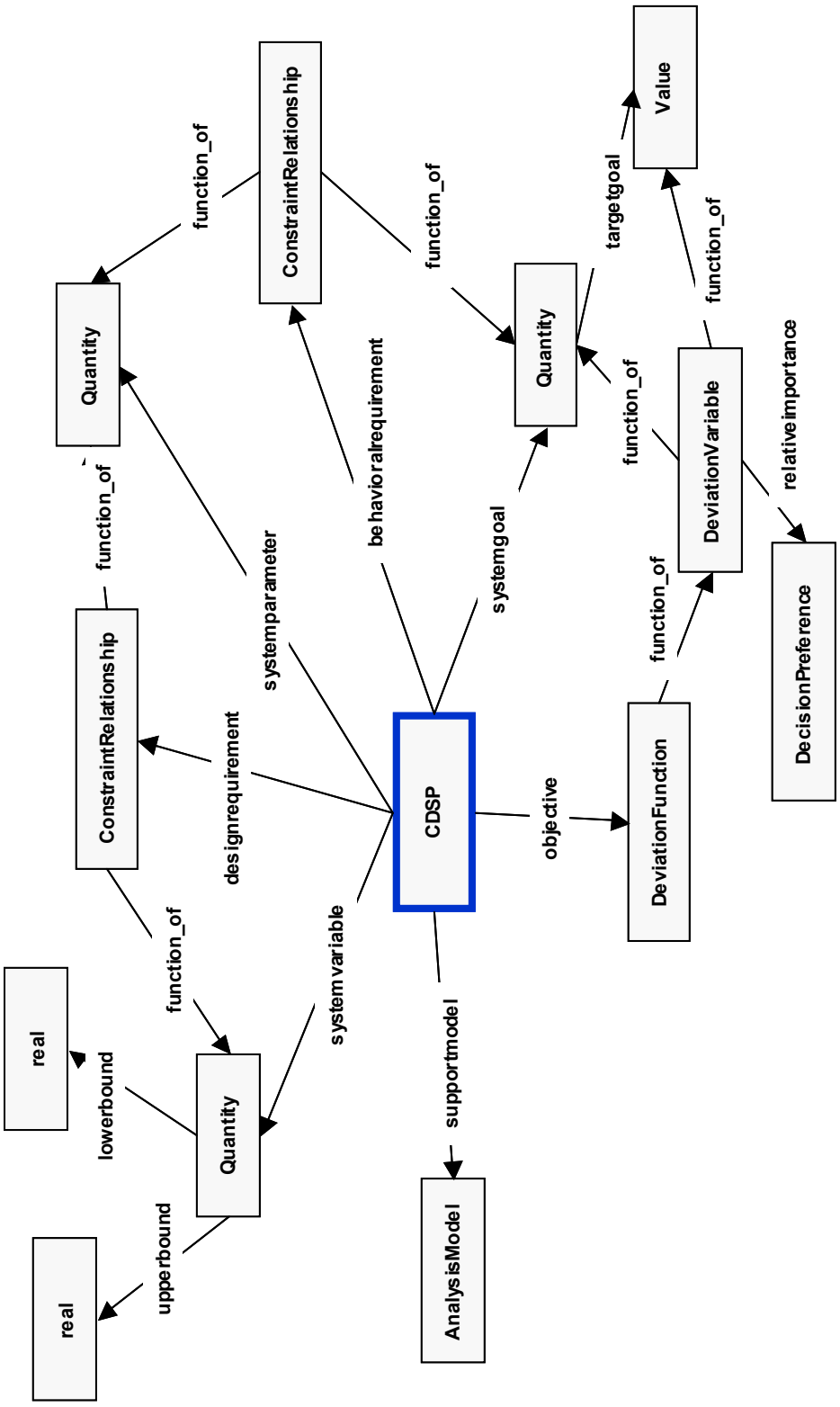


Figure 4-10: Graphical representation of cDSP concept

The cDSP information model captures the relationships between design variables, design parameters, system goals, the deviation objective function, design requirements, and analysis support models. The cDSP information model is specified using 12 properties (10 object properties and 2 data type properties) and nine concepts (7 object concepts and 2 data types). In comparison to the flat structure presented in Table 4-5, the graphical information representation enables designers to capture and visualize the complex information relationships in a design decision. The information model presented in Figure 4-11 is the foundation for developing a DL representation. The DL representation provides a computer-processible form of the cDSP and enables decision makers to model engineering decisions using an established vocabulary. The DL representation of the cDSP is:

```
Class(CDSP
and(
  (∀ designvariable.(Quantity∩(=upperbound 1)∩(=lowerbound 1)))
  (∃designvariable.(Quantity∩(=upperbound 1)∩(=lowerbound 1)))
  (≥ designvariable 1)

  (∀ designparameter.Quantity)

  (∃ hassupportmodel.AnalysisModel)
  (∀ hassupportmodel.AnalysisModel)
  (≥ hassupportmodel 1)

  (∀ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value)))
  (∃ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value)))
  (≥ systemgoal 1)
  (∃ objective.(DeviationFunction ∩
    (∃ function_of.(DeviationVariable ∩
       ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
```

```
      (= relativeimportance 1)) ∩
      ((∃ function_of.Quantity) ∩
      (∃ function_of.Value) ∩
      (∀ function_of.(Quantity∪Value))) ∩
    (∀ function_of.(DeviationVariable ∩
      ((∃ relativeimportance.RelativeImportance) ∩
      (∀ relativeimportance.RelativeImportance)∩
      (= relativeimportance 1)) ∩
      ((∃ function_of.Quantity) ∩
      (∃ function_of.Value) ∩
      (∀ function_of.(Quantity∪Value))))
  (∀objective.(DeviationFunction ∩
    (∃ function_of.(DeviationVariable ∩
      ((∃ relativeimportance.RelativeImportance) ∩
      (∀ relativeimportance.RelativeImportance)∩
      (= relativeimportance 1)) ∩
      ((∃ function_of.Quantity) ∩
      (∃ function_of.Value) ∩
      (∀ function_of.(Quantity∪Value))) ∩
    (∀ function_of.(DeviationVariable ∩
      ((∃ relativeimportance.RelativeImportance) ∩
      (∀ relativeimportance.RelativeImportance)∩
      (= relativeimportance 1)) ∩
      ((∃ function_of.Quantity) ∩
      (∃ function_of.Value) ∩
      (∀ function_of.(Quantity∪Value))))
  (= objective 1)

  (∀ behavioralrequirement.ConstraintRelationship)
  (∀ designrequirement.ConstraintRelationship))))
```

The CDSP concept is described in prose as the following:

"A cDSP has at least one design variable, where all design variables are quantities that have one lower and one upper bound ***and*** *all design parameters are quantities* ***and*** *has at least one support model, where all support models are analysis models* ***and*** *has at least one system goal, where all system goals are quantities* ***and*** *where all systems goals have at least a target goal that is a value, where all target goal are values* ***and*** *has exactly one objective, where the objective is a deviation function* ***and*** *the deviation*

function is a function of at least one deviation variables, where the deviation function is a function of only deviation variables ***and*** *where all deviation variables have exactly one relative importance, where the relative importance is a decision preference* ***and*** *where a deviation variable is a function a quantity and a function of a value* ***and*** *all behavior requirements are constraint relationships* ***and*** *all design requirements are constraint relationships"*

The `CDSP` concept is described using $\mathcal{ALEN}$ DL, which is the basic description logic language ($\mathcal{ALC}$) with the addition of unqualified number restrictions ($\mathcal{N}$). The qualified number restrictions are used to specify the cardinality of properties in the concept definition. Examples of qualified number restrictions in the cDSP concept include:

```
designvariable.(Quantity∩(=upperbound 1)∩(=lowerbound 1)))
(≥ designvariable 1)
(≥ hassupportmodel 1)
(≥ systemgoal 1)
```

The information base for the compromise decision support problem is developed by committing the concept description. In this context, an information base is a generalization of databases and knowledge bases that emphasize the representation and storage of information symbols [100]. The information base consists of only the cDSP concept, resulting in a simple organization structure (see Figure 4-11).

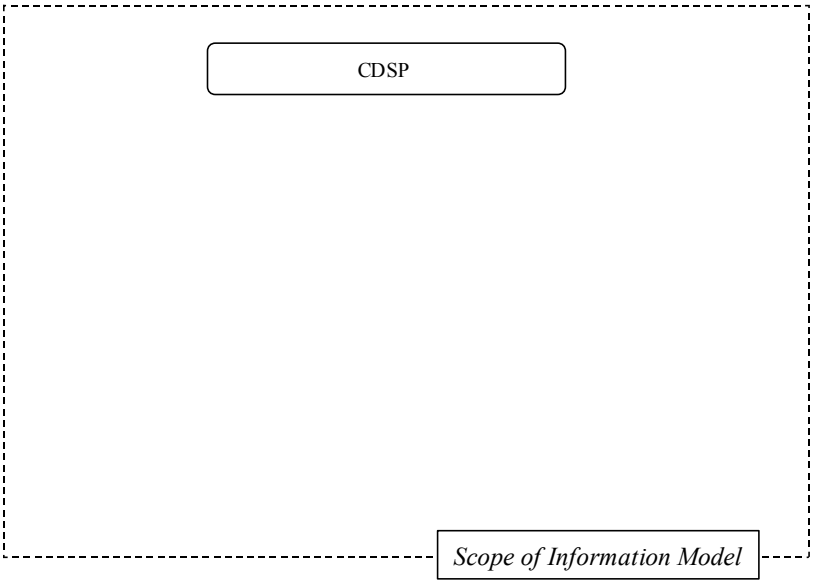


Figure 4-11: Decision construct information model – State 1

The information model presented in Figure 4-11 is not particularly interesting because it only contains a single concept. However, it is presented as a datum through which the addition, modification, and organization of other specified concepts are related.

The first modification to the knowledge base and decision vocabulary tests the claims made in Chapter 3 that DL provides an extensible and robust means for developing information models. In the following sections, examples are provided to illustrate the robustness and extensibility of the vocabulary for specifying new concepts.

4.5.2 Alternative cDSP Concept Information Representation

To demonstrate the robustness and extensibility of DL and the cDSP vocabulary an alternative cDSP concept definition is specified. The `AlternativeCDSP` specifies the same information structure and content as the `CDSP` concept. The `AlternativeCDSP` is defined as:

```
Class(AlternativeCDSP
and(
  (∀ designvariable.CDSPDesignVariable)
  (∃ designvariable.CDSPDesignVariable)
  (≥ designvariable 1)

  (∀ designparameter.Quantity)

  (∃ hassupportmodel.AnalysisModel)
  (∀ hassupportmodel.AnalysisModel)
  (≥ hassupportmodel 1)

  (∃ systemgoal.CDSPSystemGoal
  (∀ systemgoal.CDSPSystemGoal
 (≥ systemgoal 1)

  (∃ objective.(CDSPDeviationFunction)
  (∀ objective.(CDSPDeviationFunction)
  (≥ objective 1)

  (∀ behavioralrequirement.ConstraintRelationship)
  (∀ designrequirement.ConstraintRelationship))))
```

where CDSPGoal is defined as:

```
Class(CDSPSystemGoal
  and(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value)))
```

where CDSPDesignVariable is defined as:

```
Class(CDSPDesignVariable
  and(
  Quantity
  (=upperbound 1)
  (=lowerbound 1)))
```

where CDSPDeviationFunction is defined as:

```
Class(CDSPDeviationFunction
  and((∃ function_of.DeviationVariable)
      (∀ function_of.DeviationVariable)))
```

and where `CDSPDeviationVariable` is defined as:

```
Class(CDSPDeviationVariable
  and((∃ relativeimportance.RelativeImportance)
      (∀ relativeimportance.RelativeImportance)
      (= relativeimportance 1)
      (∃ function_of.Quantity)
      (∃ function_of.Value)
      (∀ function_of.(Quantity∪Value))))
```

The `AlternativeCDSP` is semantically equivalent to the CDSP concept. In other words, the two concepts are identical in meaning, but expressed using different intermediate concepts. Concept equivalence is determined through the DL reasoner. Concepts are equivalent if every interpretation of two concepts. Concepts are equivalent if $(C \cap \neg D)$ and $(\neg C \cap D)$ are unsatisfiable. The knowledge base expressed in Figure 4-11 is expanded based on the equivalent definition (see Figure 4-12).

Identification of equivalent concepts is important in engineering design because it (1) enables information model developers to reduce the complexity in the information models and (2) enables designers to retrieve concepts from the information model that are semantically equivalent. The example presented in Figure 4-16 is quite simple, but captures the essence of determining concept equivalence. Additionally, concept equivalence has been shown to be complete and sound for the description language used in this research [16]. Thus, concept equivalence for complex concepts is supported because reasoning algorithms have been proven to be sound and complete for a variety of

description languages. In the following two sections, the CDSP concept is specialized through the addition of DL constructs and axioms.

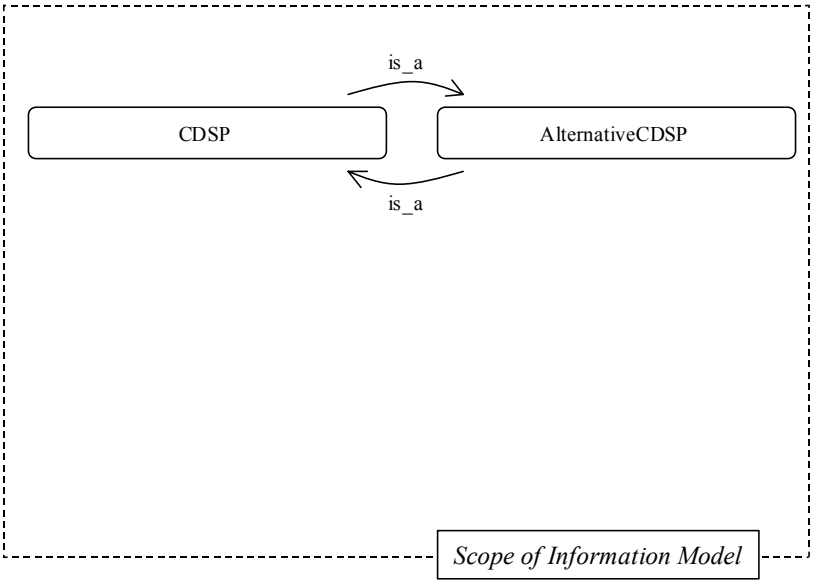


Figure 4-12: Decision construct information model – State 2

4.5.3 Multi-Goal and Single-Goal cDSP Information Representations

Additional concepts are specified using the vocabulary. Concepts are created that are specializations of the cDSP concept with additional restrictions specified in the concept definition. Multi-goal and single-goal cDSP are created by modifying the qualified number restrictions. A multi-goal cDSP is defined as a cDSP that must have more than one design goal that must be achieved. Foreshadowing to the examples presented in Chapter 5 and 6, examples of multi-goal cDSPs include:

- Design of a beam in which the weight of the beam and the stress in the beam must be balanced
- Design of a heat sink in which the thermal resistance of the heat sink must be balanced against the size of the heat sink

The MultiGoalCDSP concept is defined by changing the unqualified number restriction in the CDSP concept. The expression in the CDSP concept is altered from (≥ systemgoal 1)) to (≥ systemgoal 2)). The DL abstract syntax for the multi-objective cDSP is defined as:

```
Class(MultiGoalCDSP
and(
  (∀ designvariable.(Quantity∩(=upperbound 1)∩(=lowerbound 1)))
  (∃designvariable.(Quantity∩(=upperbound 1)∩(=lowerbound 1)))
  (≥ designvariable 1)

  (∀ designparameter.Quantity)

  (∃ hassupportmodel.AnalysisModel)
  (∀ hassupportmodel.AnalysisModel)
  (≥ hassupportmodel 1)

  (∀ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value)))
  (∃ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value)))
 (≥ systemgoal 2)

  (∃ objective.(DeviationFunction ∩
    (∃ function_of.(DeviationVariable ∩
       ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩
       ((∃ function_of.Quantity) ∩
       (∃ function_of.Value) ∩
       (∀ function_of.(Quantity∪Value))) ∩
    (∀ function_of.(DeviationVariable ∩
       ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩
       ((∃ function_of.Quantity) ∩
       (∃ function_of.Value) ∩
       (∀ function_of.(Quantity∪Value))))
```

```
(∀objective.(DeviationFunction ∩
  (∃ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))) ∩
  (∀ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))))
(= objective 1)

(∀ behavioralrequirement.ConstraintRelationship)
(∀ designrequirement.ConstraintRelationship))))
```

The `MultiGoalCDSP` concept is described in prose as the following:

"A multi-goal cDSP has at least one design variable, where all design variables are quantities that have one lower and one upper bound ***and*** *all design parameters are quantities* ***and*** *has at least one support model, where all support models are analysis models* ***and*** *has at least two system goals, where all system goals are quantities* ***and*** *where all systems goals have at least a target goal that is a value, where all target goal are values* ***and*** *has exactly one objective, where the objective is a deviation function* ***and*** *the deviation function is a function of at least one deviation variables, where the deviation function is a function of only deviation variables* ***and*** *where all deviation variables have exactly one relative importance, where the relative importance is a decision preference* ***and*** *where a deviation variable is a function a quantity and a function of a value* ***and*** *all behavior requirements are constraint relationships* ***and*** *all design requirements are constraint relationships"*

The conceptual information model is changed to Figure 4-13 through the addition of the `MultiGoalCDSP`.

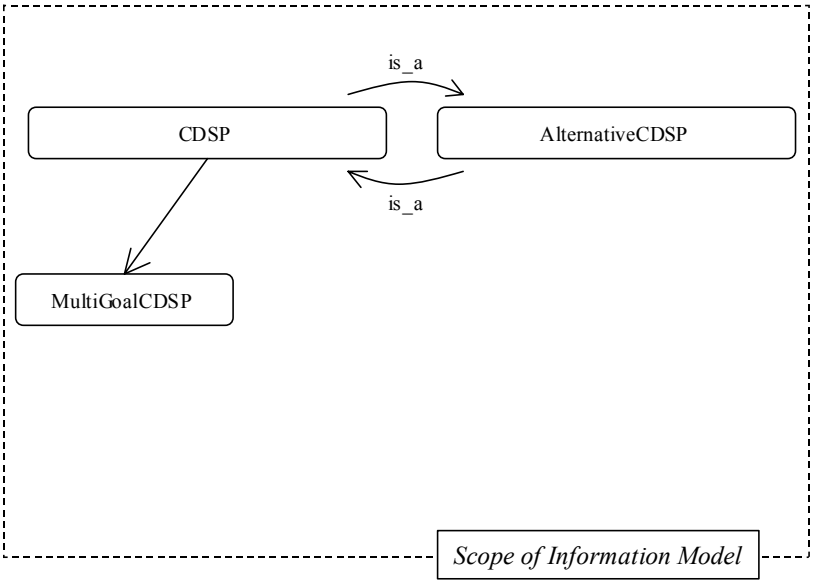


Figure 4-13: Decision construct information model – State 3

Additionally, a single-goal compromise decision support problem is represented using the DL language. The single-goal cDSP is a specialization of the general cDSP construct, and thus is a subclass. The single-goal cDSP concept definition is specified as:

```
Class(SingleGoalCDSP
and(
  (∀ designvariable.(Quantity∩(=upperbound 1)∩(=lowerbound 1)))
  (∃designvariable.(Quantity∩(=upperbound 1)∩(=lowerbound 1)))
  (≥ designvariable 1)
  (∀ designparameter.Quantity)

  (∃ hassupportmodel.AnalysisModel)
  (∀ hassupportmodel.AnalysisModel)
  (≥ hassupportmodel 1)
  (∀ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value)))
  (∃ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
```

```
  (∀ targetsystembehavior.Value)))
(= systemgoal 1)


(∃ objective.(DeviationFunction ∩
  (∃ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))) ∩
  (∀ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))))
(∀objective.(DeviationFunction ∩
  (∃ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))) ∩
  (∀ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))))
(= objective 1)
(∀ behavioralrequirement.ConstraintRelationship)
(∀ designrequirement.ConstraintRelationship))))
```

The `SingleGoalCDSP` concept is described in prose as the following:

"A single-goal cDSP has at least one design variable, where all design variables are quantities that have one lower and one upper bound ***and*** *all design parameters are quantities* ***and*** *has at least one support model, where all support models are analysis*

*models **and** has exactly one system goal, where all system goals are quantities **and** where all systems goals have at least a target goal that is a value, where all target goal are values **and** has exactly one objective, where the objective is a deviation function **and** the deviation function is a function of at least one deviation variables, where the deviation function is a function of only deviation variables **and** where all deviation variables have exactly one relative importance, where the relative importance is a decision preference **and** where a deviation variable is a function a quantity and a function of a value **and** all behavior requirements are constraint relationships **and** all design requirements are constraint relationships"*

The information model reflects that modification through the addition of the single goal cDSP (see Figure 4-14). To this point, modifications and additions to the information base are completed in two ways: (1) the first is creating semantically equivalent concepts and (2) creating specializations based on DL constructs and number restrictions. The three examples illustrate how specific concepts can be defined based on more restrictive statement in the concept definitions. The creation of hierarchical relationships in the previous examples is accomplished by specialized concept definitions, not by explicitly defining subclass/superclass relationships. In traditional database and information modeling development the hierarchies are created manually. While this is not straightforward in traditional database implementation is can be down by creating new entities (and table). However, a more complex modification to the knowledge base involves creating subclasses and correctly identifying what the concepts new concepts subsumed.

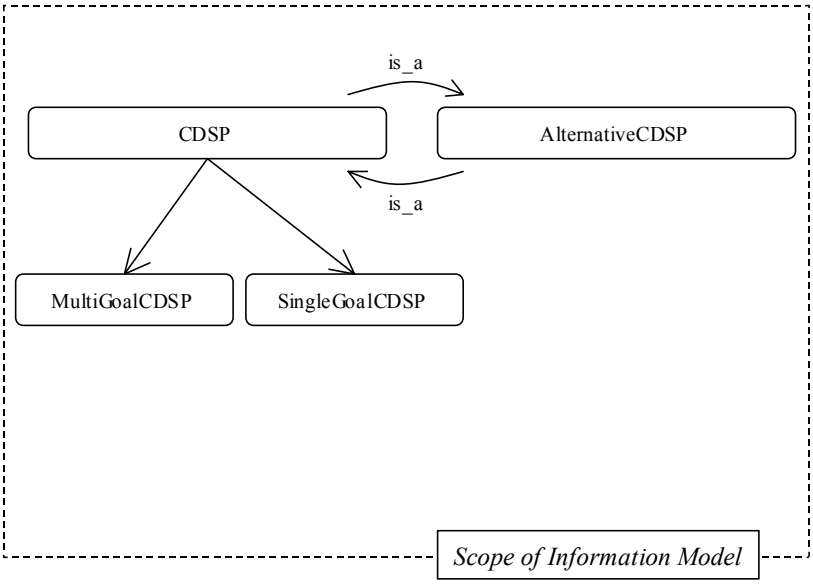


Figure 4-14: Decision construct information model – State 4

To this point, concepts are defined using the base vocabulary. However, as stated in Chapter 3, the longevity and dynamic nature of engineering information cause problem in the evolution and extensibility of information models developed using traditional database formalisms. In the following section several examples are presented to illustrate how DL enable the base vocabulary to be extended while minimize the propagation of change throughout the information base.

4.5.4 Extensions to the Design Decision Information Model

Several examples are presented in the previous section that illustrate how different types of design decisions can be represented using the established vocabulary. However, the question arises, *Given the basic vocabulary of concepts and properties, how can it be extended to enable modeling of concepts that fall outside of the current language?* As discussed in Chapter 3, engineering information is dynamic throughout the realization process. Hence, the information models used to support the product realization process

must also be dynamic. The information model developed at state *i* may be complete and comprehensive. However, as the modeling needs change and additional information is identified he model must be able to changed and extended easily.

Three examples are presented to illustrate the extensibility and robustness of the basic language. In this context, extensibility refers to the ability to change and robustness refers to the propagation and stability of the changes in the information model. These examples include compromise decision support problems that take into account robust design principles including, Type I and Type II robustness. The details of robust compromise decision support problem are found in [36; 37]. Robust design is an approach for improving product quality by reducing the sensitivity to variations. The reduction is sought without removing the sources of the variability. A robust design is a system that can tolerate variation from the external environment or internal design specification without suffering from major performance degradation. The underlying principle of robust design is to determine superior solutions to design problems through minimizing the effects of variation, without eliminating their causes. There are two categories of robust design problems classified in [36; 37]. Both simultaneously bring the mean system performance to a target and minimize performance variation; however, the sources of the variation are different (see Figure 4-15 and Figure 4-16).

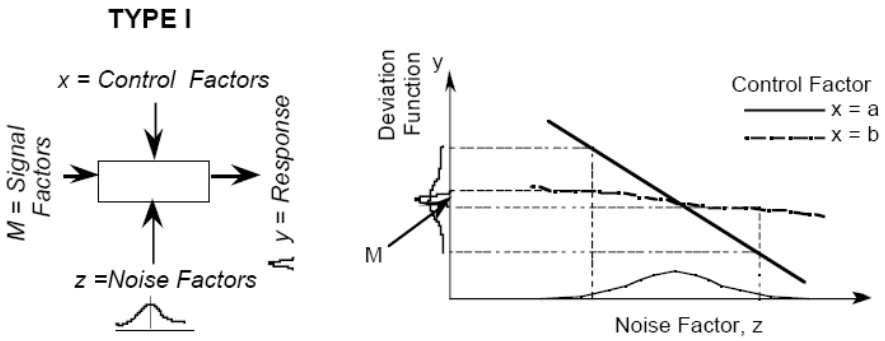


Figure 4-15: Type I Robust Design [36]

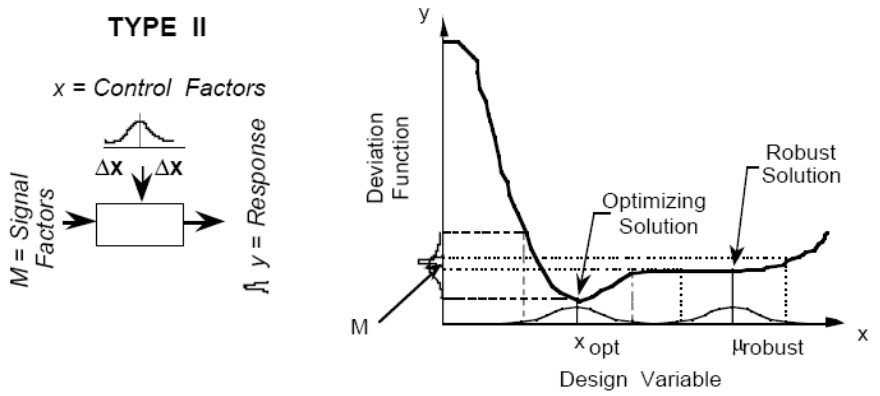


Figure 4-16: Type II Robust Design [36]

- Type I – minimizing variations in performance caused by variations in noise factors (uncontrollable parameters)
- Type II – minimizing variations in performance caused by variations in control factors (design variables)

Chen and coauthors proposed solving robust design as a multi-objective decision problem of bringing the mean to target and minimizing the variation of the response. In this approach both control and noise factors are considered as potential sources of

variation, and constraints are modeled in a worst case scenario formulation to ensure feasibility. While all the details are not presented, the cDSP is reformulated to include variation on the design variable, variation on design / noise parameters, and target mean and variance of the system goals. Thus, additional information is needed to formulate and subsequently solve a robust cDSP, the additional information is shown in italics in Table 4-6.

Table 4-6: Keywords and descriptors for Robust Compromise Decision Support Problem

Keywords	Descriptors
Given	System **design parameters** and constants **Analysis models** necessary for evaluating the goals, constraints and bounds, and the deviation function ***Mean and variation on design variables and noise parameters*** ***Target variance for design goals***
Find	System **design variables** and deviation variables
Satisfy	**Systems goals**, **constraints** and **bounds**
Minimize	A deviation function ***as a function of the overall mean and variance of each of the system goals***

In order to represent the robust cDSP formulation, three properties were added to the vocabulary. The expressivity of the language remains and the complexity of the language was changed to $\mathcal{ALCON}$ (see Table 4-7).

The base vocabulary is extended to enable modeling in the information associated with robust design decisions. In the following sections, the vocabulary is exercised for representing three different types of robust design decisions.

Table 4-7: Additional properties added to language for modeling robust decisions

Property	Definition
noiseparameter	Represents the parameters the designer does not control that are considered to be a noise factor. Noise parameters are parameters that have a mean value and a variance value. D: CDSP R: Quantity
variation	A statistical variation value for a system design variable or parameter D: Quantity R: Value
targetvariance	Specifies the target variance of interest for a design goal. D: Quantity R: Value

4.5.4.1 Type I -II Robust cDSP Information Representation

In general, design decisions often involve uncertainty, or imperfections, in design variables (control factors) and systems goals (response) and noise parameters must be taken into consideration The robust cDSP decision formulation proposed by Chen and co-authors [34] models uncertainty in the design decision as a combination of a statistical mean and variance. The robust cDSP formulation enables designers to mathematically model engineering design decisions in which uncertainty is taken into consideration. Specifically, the overall objective in the robust cDSP formulation is to maximize the deviation of the mean and minimize the deviation from the target variance of the system goals. In the Type I-II Robust cDSP, the decision maker takes into account the variance on system design variables, noise parameters, and target variance for system goals. The information associated with the cDSP is modified to reflect the formulation of Type I-II Robust cDSP in Figure 4-17

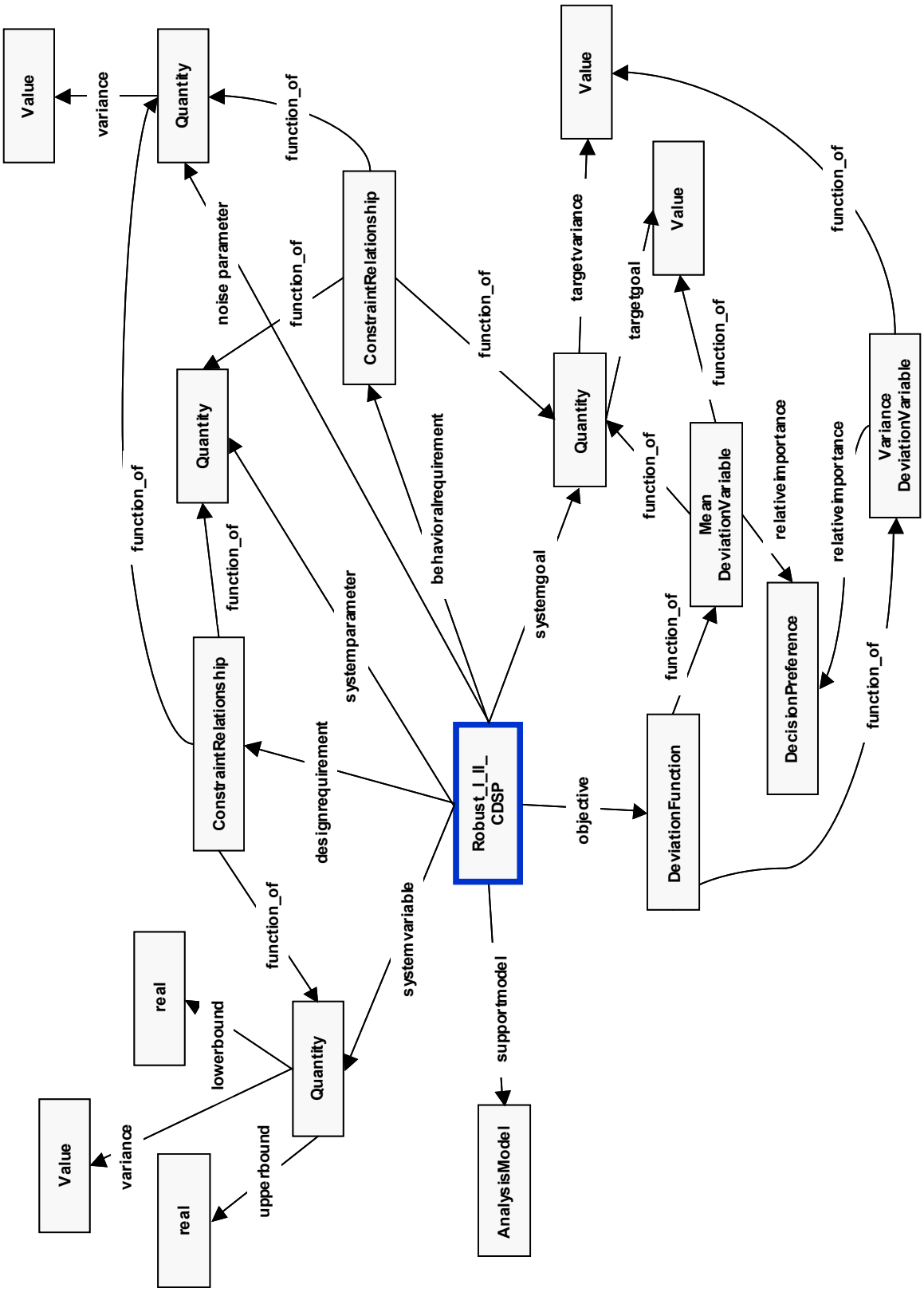


Figure 4-17: Graphical representation of Type I-II Robust cDSP concept

As illustrated in Figure 4-19, the Type I-II Robust cDSP includes the targetvariance property on systemgoal.Quantity. Additionally, the noiseparameter.Quantity property is added to the Robust I-II concept definition and the DeviationVariable concept is stated as a function of both the noiseparameter.Quantity and the targetvariance.Value concepts. Finally, the systemvariable.Quantity concept is defined to have a variance property that takes a Value. The Type I-II Robust compromise decision support problem concept is defined in DL as:

```
Class(Robust_I_II_CDSP
and(
  (∃ designvariable.(Quantity ∩
       (= has_upper_bound cardinality 1) ∩
       (= has_lower_bound cardinality 1) ∩
       (∃ variance.Value) ∩
       (∀ variance.Value) ∩
       (=variance 1)))
  (∀ designvariable.(Quantity ∩
       (= has_upper_bound cardinality 1) ∩
       (= has_lower_bound cardinality 1) ∩
       (∃ variance.Value) ∩
       (∀ variance.Value) ∩
       (=variance 1)))
  (≥ designvariable 1)

  (∀ designparameter.Quantity)
  (∀ noiseparameter.(Quantity∩
    (∃ variation.Value) ∩
    (∀ variation.Value) ∩
    (= variation 1)))

  (∃ hassupportmodel.AnalysisModel)
  (∀ hassupportmodel.AnalysisModel)
  (≥ hassupportmodel 1)

  (∀ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value) ∩
    (∃ targetsystembehaviorvariance.Value) ∩
```

```
  (∀ targetsystembehaviorvariance.Value) ∩
  (= targetsystembehaviorvariance 1)))
(∃ systemgoal.(Quantity ∩
  (= targetsystembehavior 1) ∩
  (∃ targetsystembehavior.Value) ∩
  (∀ targetsystembehavior.Value) ∩
  (∃ targetsystembehaviorvariance.Value) ∩
  (∀ targetsystembehaviorvariance.Value) ∩
  (= targetsystembehaviorvariance 1)))
(≥ systemgoal 1)

(∃ objective.(DeviationFunction ∩
  (∃ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))) ∩
  (∀ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((= function_of 2) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))))
(∀objective.(DeviationFunction ∩
  (∃ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))) ∩
  (∀ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((= function_of 2) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))))
(= objective 1)

(∀ behavioralrequirement.ConstraintRelationship)
(∀ designrequirement.ConstraintRelationship))))
```

The `Robust_I_II_CDSP` concept is described in prose as the following:

"A type I robust cDSP has at least one design variable, where all design variables are quantities that have one lower and one upper bound, and one variance value ***and*** *all design parameters are quantities* ***and*** *all noise parameters are quantities that have a variance* ***and*** *has at least one support model, where all support models are analysis models* ***and*** *has at least two system goals, where all system goals are quantities* ***and*** *where all systems goals have at least a target goal that is a value, where all target goal are values* ***and*** *where all systems goals have at exactly one target variance that is a value* ***and*** *has exactly one objective, where the objective is a deviation function* ***and*** *the deviation function is a function of at least one deviation variables, where the deviation function is a function of only deviation variables* ***and*** *where all deviation variables have exactly one relative importance, where the relative importance is a decision preference* ***and*** *where a deviation variable is a function a quantity and a function of a value* ***and*** *all behavior requirements are constraint relationships* ***and*** *all design requirements are constraint relationships"*

As discussed above, The `Robust_I_II_cDSP` concept is a specialization of the `CDSP` concept. The `Robust_I_II_CDSP` concept requires the specification of additional information including, design parameters identified as uncontrollable noise parameters, variance values specified for design variables, and target variance value defined for the system goals. The Robust_I_II_CDSP concept results in a modification to the information model illustrated in Figure 4-13.

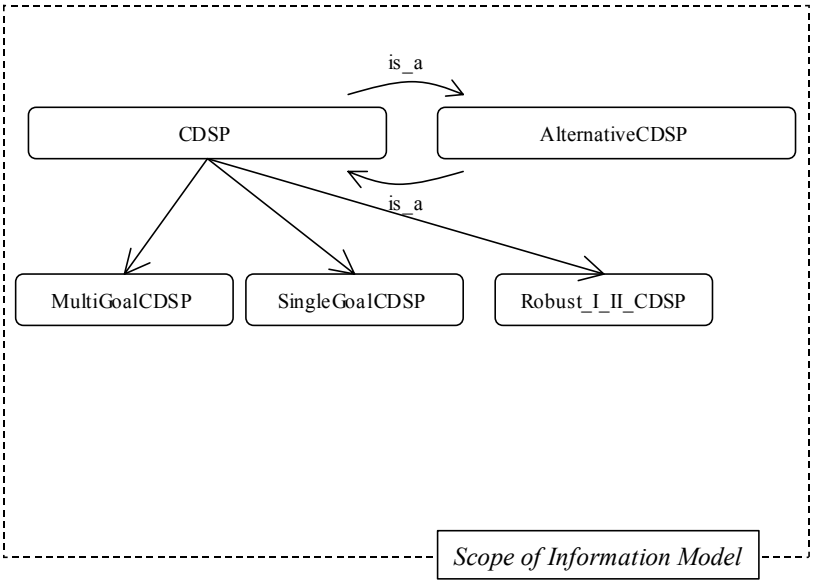


Figure 4-18: Decision construct information model – State 5

According to the concept definition, the Robust_I_II_CDSP requires the designer to specify additional information. Thus, the Robust_I_II_CDSP concept is a specialization of the general cDSP concept (i.e., The information associated with a robust decision is sufficient to enable a designer to model a general cDSP). As the name implies, and as previously discussed, the Robust I_II_ CDSP concept enables the designer to model two types of robust design considerations, namely Type I and Type II robust decisions. In the following sections, concept definitions are developed for Type I and Type II design decision individually. The rationale for specifying the robust design decisions concept definitions in the prescribed order is to illustrate how the reasoning algorithms are used to dynamically organize concepts and ensure consistency in the information model.

4.5.4.2 Type I Robust cDSP Information Representation

As previously stated, the objective of the Type I Robust cDSP formulation is to minimize the deviation of the variance from the target variance and maximize the mean (target) from the target of the system goals, given the presence of noise parameters associated with the design decision. The Robust_I_CDSP concept is more specific than the general cDSP concept, but is a generalization of the Robust_I_II_CDSP. The Robust_I_CDSP concept only takes into account the variance associated with noise parameters and a target variance on the systems goals. The type I robust cDSP information model is illustrated graphically in Figure 4-19.

As illustrated in Figure 4-19, the Type I Robust cDSP includes the targetvariance property on the `systemgoal.Quantity` assertion. Additionally, the `noiseparameter.Quantity` property is added to the Robust I concept definition and the `DeviationVariable` concept is stated as a function of both the `noiseparameter.Quantity` and the `targetvariance.Value` concepts.

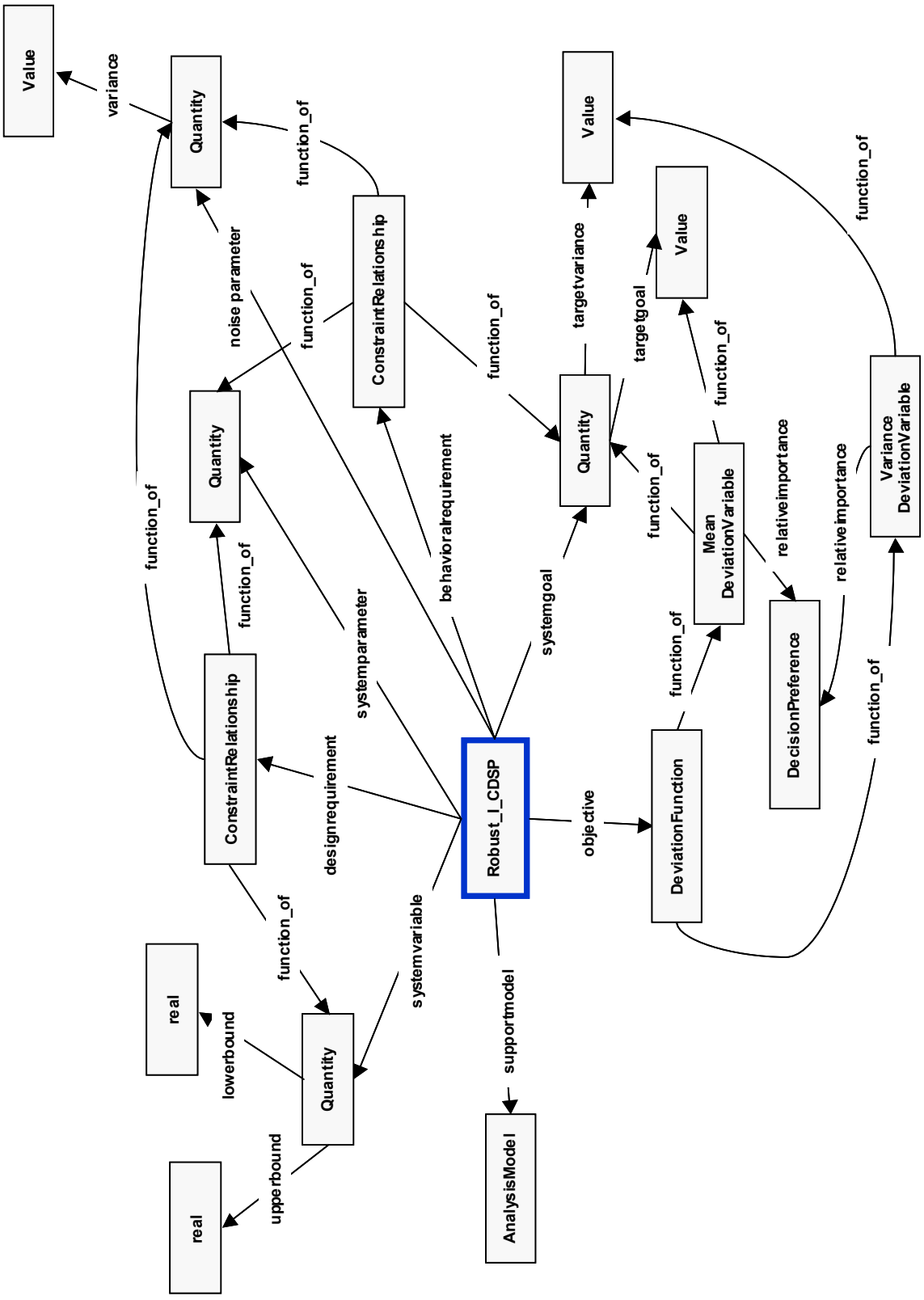


Figure 4-19: Graphical representation of Type I Robust cDSP concept

The Type I Robust compromise decision support problem concept is defined in DL as:

```
Class(Robust_I_CDSP
and(
  (∀ designvariable.(Quantity∩(=upperbound 1)∩(=lowerbound 1)))
  (∃designvariable.(Quantity∩(=upperbound 1)∩(=lowerbound 1)))
  (≥ designvariable 1)

  (∀ designparameter.Quantity)
  (∀ noiseparameter.(Quantity∩
  (∃ variation.Value) ∩
  (∀ variation.Value) ∩
  (= variation 1)))

  (∃ hassupportmodel.AnalysisModel)
  (∀ hassupportmodel.AnalysisModel)
  (≥ hassupportmodel 1)

  (∀ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value) ∩
    (∃ targetsystembehaviorvariance.Value) ∩
    (∀ targetsystembehaviorvariance.Value) ∩
    (= targetsystembehaviorvariance 1)))
  (∃ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value) ∩
    (∃ targetsystembehaviorvariance.Value) ∩
    (∀ targetsystembehaviorvariance.Value) ∩
    (= targetsystembehaviorvariance 1)))
  (≥ systemgoal 1)
  (∃ objective.(DeviationFunction ∩
    (∃ function_of.(DeviationVariable ∩
       ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩

       ((∃ function_of.Quantity) ∩
       (∃ function_of.Value) ∩
       (∀ function_of.(Quantity∪Value))) ∩
    (∀ function_of.(DeviationVariable ∩
       ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩
       ((∃ function_of.Quantity) ∩
```

```
      (∃ function_of.Value) ∩
      (∀ function_of.(Quantity∪Value))))
  (∀objective.(DeviationFunction ∩
    (∃ function_of.(DeviationVariable ∩
      ((∃ relativeimportance.RelativeImportance) ∩
      (∀ relativeimportance.RelativeImportance)∩
      (= relativeimportance 1)) ∩
      ((∃ function_of.Quantity) ∩
      (∃ function_of.Value) ∩
      (∀ function_of.(Quantity∪Value))) ∩
    (∀ function_of.(DeviationVariable ∩
      ((∃ relativeimportance.RelativeImportance) ∩
      (∀ relativeimportance.RelativeImportance)∩
      (= relativeimportance 1)) ∩
      ((∃ function_of.Quantity) ∩
      (∃ function_of.Value) ∩
      (∀ function_of.(Quantity∪Value))))
  (= objective 1)

  (∀ behavioralrequirement.ConstraintRelationship)
  (∀ designrequirement.ConstraintRelationship))))
```

The Robust_I_CDSP concept is described in prose as the following:

"A type I robust cDSP has at least one design variable, where all design variables are quantities that have one lower and one upper bound ***and*** *all design parameters are quantities* ***and*** *all noise parameters are quantities that have a variance* ***and*** *has at least one support model, where all support models are analysis models* ***and*** *has at least two system goals, where all system goals are quantities* ***and*** *where all systems goals have at least a target goal that is a value, where all target goal are values* ***and*** *where all systems goals have at exactly one target variance that is a value* ***and*** *has exactly one objective, where the objective is a deviation function* ***and*** *the deviation function is a function of at least one deviation variables, where the deviation function is a function of only deviation variables* ***and*** *where all deviation variables have exactly one relative importance, where the relative importance is a decision preference* ***and*** *where a deviation variable is a*

function a quantity and a function of a value ***and*** *all behavior requirements are constraint relationships* ***and*** *all design requirements are constraint relationships"*

The Robust_I_CDSP is a specialization of the cDSP in that it requires design parameters to be identified as an uncontrollable noise parameter and a target variance to be specified for the system goals. However, the Robust_I_CDSP is a generalization of the Robust_I_II_CDSP. The addition of the Robust_I_CDSP concept definition results in a modification to the information model illustrated in Figure 4-20.

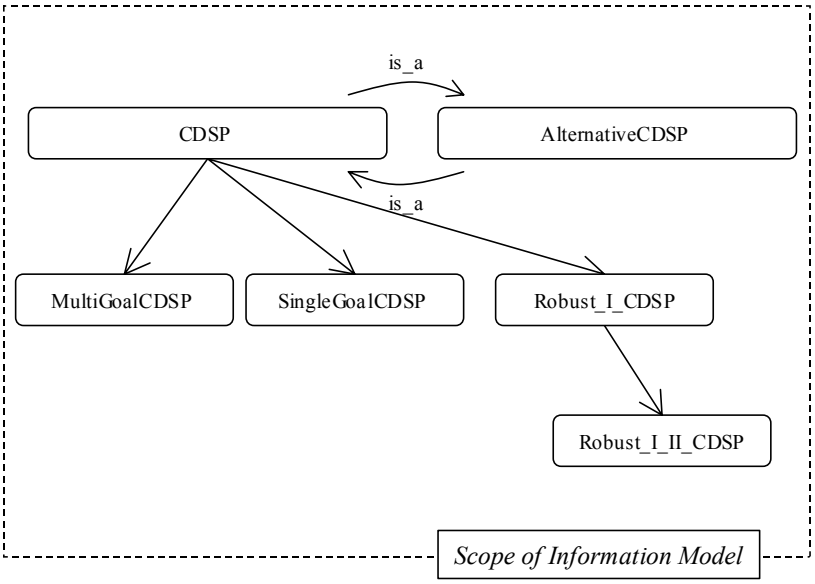


Figure 4-20: Decision construct information model – State 6

The information model and concept classification (super/sub class) relationships are maintained as the Robust_I_CDSP concept definition is specified. As illustrated in Figure 4-20, the Robust_I_CDSP is added to the information model and is simultaneously subsumed by the CDSP concept and subsumes the Robust_I_II_CDSP concept. In the following section, the information model and concept specification for type II robust

design decisions is presented. The information model is similar (almost identical) to the Type_I_Robust_CDSP developed in this section.

4.5.4.3 Type II Robust cDSP Information Representation

Type II robust design is similar to Type I robust design formulation. However, in the Type II robust compromise decision support problem formulation the objective is to minimize the variance and maximize the mean of the response given that there is variance in the designer variables. The graphical information model for the Type II Robust cDSP is illustrated graphical in Figure 4-21.

As illustrated in Figure 4-21, the Type II Robust cDSP include the `variance` property on the `systemvariable.Quantity` and `targetvariance` on property on the `systemgoal.Quantity` assertion. Additionally, the `DeviationVariable` concept is a function of both the `Quantity` and the `targetvariance.Value` statements.

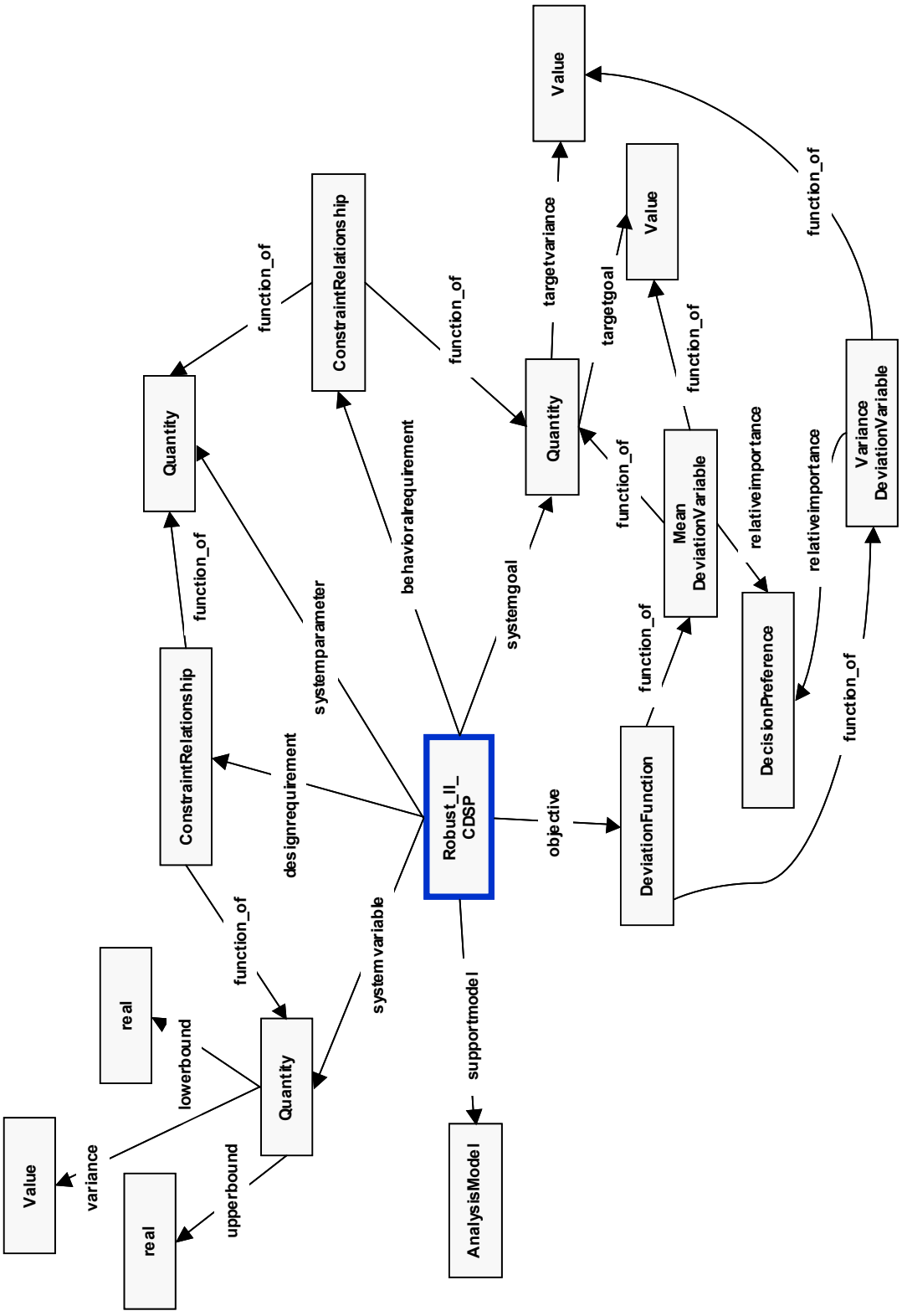


Figure 4-21: Graphical representation of Type II Robust cDSP

The Type II Robust cDSP is defined in DL as:

```
Class(Robust_II_CDSP
and(
    (∃ designvariable.(Quantity ∩
       (= has_upper_bound cardinality 1) ∩
       (= has_lower_bound cardinality 1) ∩
       (∃ variance.Value) ∩
       (∀ variance.Value) ∩
       (=variance 1)))
    (∀ designvariable.(Quantity ∩
       (= has_upper_bound cardinality 1) ∩
       (= has_lower_bound cardinality 1) ∩
       (∃ variance.Value) ∩
       (∀ variance.Value) ∩
       (=variance 1)))
    (≥ designvariable 1)

  (∀ designparameter.Quantity)

  (∃ hassupportmodel.AnalysisModel)
  (∀ hassupportmodel.AnalysisModel)
  (≥ hassupportmodel 1)

  (∀ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value) ∩
    (∃ targetsystembehaviorvariance.Value) ∩
    (∀ targetsystembehaviorvariance.Value) ∩
    (= targetsystembehaviorvariance 1)))
  (∃ systemgoal.(Quantity ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.Value) ∩
    (∀ targetsystembehavior.Value) ∩
    (∃ targetsystembehaviorvariance.Value) ∩
    (∀ targetsystembehaviorvariance.Value) ∩
    (= targetsystembehaviorvariance 1)))
  (≥ systemgoal 1)
```

```
(∃ objective.(DeviationFunction ∩
  (∃ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))) ∩
  (∀ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))))
(∀objective.(DeviationFunction ∩
  (∃ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))) ∩
  (∀ function_of.(DeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.Quantity) ∩
     (∃ function_of.Value) ∩
     (∀ function_of.(Quantity∪Value))))
(= objective 1)

(∀ behavioralrequirement.ConstraintRelationship)
(∀ designrequirement.ConstraintRelationship))))
```

The Robust_II_CDSP concept is described in prose as the following:

"A type II robust cDSP has at least one design variable, where all design variables are quantities that have one lower, one upper bound, and a variance value ***and*** *all design parameters are quantities* ***and*** *has at least one support model, where all support models are analysis models* ***and*** *has at least two system goals, where all system goals are quantities* ***and*** *where all systems goals have at least a target goal that is a value, where*

all target goal are values ***and*** *where all systems goals have at exactly one target variance that is a value* ***and*** *has exactly one objective, where the objective is a deviation function* ***and*** *the deviation function is a function of at least one deviation variables, where the deviation function is a function of only deviation variables* ***and*** *where all deviation variables have exactly one relative importance, where the relative importance is a decision preference* ***and*** *where a deviation variable is a function a quantity and a function of a value* ***and*** *all behavior requirements are constraint relationships* ***and*** *all design requirements are constraint relationships"*

The `Robust_II_CDSP` concept is a specialization of the `CDSP` concept and requires that the design variables have a variance value specified to represent the uncertainty in the variable `Quantity`. Additionally, a target variance must be specified for the system goals. The modification to the decision information model is captured in Figure 4-22.

As illustrated in Figure 4-22, the Type_II_Robust_CDSP concept specification is organized in the concept hierarchy. Similar to the Type I robust design decision concept, the Type_II_Robust_CDSP concept is subsumed by the CDSP concept and subsumes the Type_I_II_Robust_CDSP concept. As a result of creating the Type_II_Robust_CDSP concept definition, the Type_I_II_Robust_CDSP concept is subsumed by the Type I and Type II concepts.

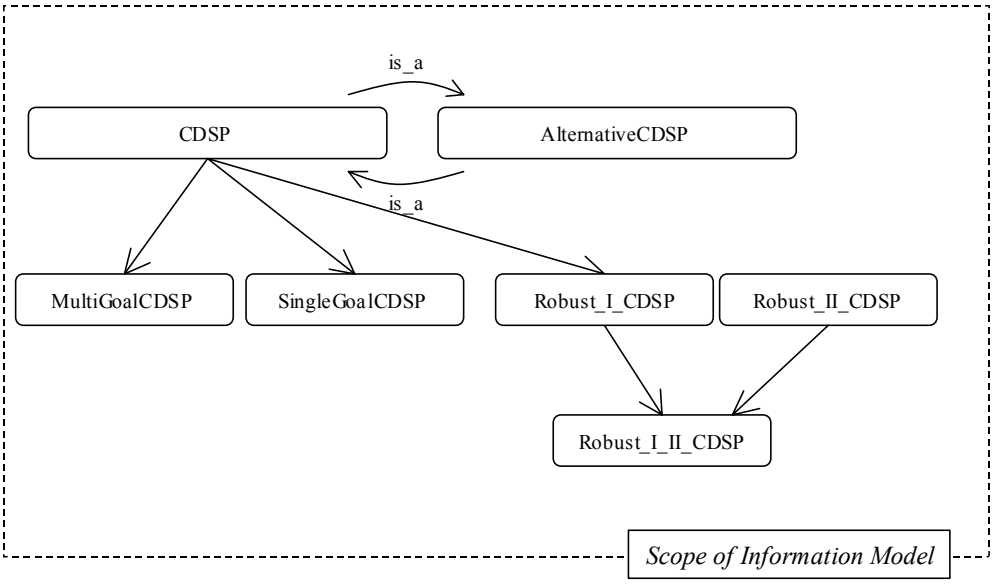


Figure 4-22: Decision construct information model – State 7

The relationships between the three different robust decision formulations is quite obvious. Clearly, the robust I-II design decision is more specific than each of the individual robust decisions. Standard DL reasoning algorithms are used to determine the implicit subclass-superclass relationships between concepts specified in the information model. Multiple inheritance is determined between the three different types of robust design decisions.

The robust design decision examples are not complex. However, the benefit of developing information models based on DL is illustrated through dynamic organization and consistency of modeling concepts. A detailed discussion and critical analysis of DL for engineering information modeling is included in Section 4.8. To this point, the vocabulary is used exclusively for specifying the information structure associated with engineering design decision. Several examples are presented that demonstrate the use of DL for developing engineering information models and how standard reasoning

algorithms enable information organization and consistency checking. In addition to representing the information associated with engineering design decisions, engineering analysis models provide insight into the behavior or response of the product. In the following section, an information model for engineering analysis models is presented.

4.6 Information Modeling of Engineering Analysis Models

The concepts and properties defined in Table 4-2 are used to represent the information associated with engineering analysis models. Engineering analysis models enable designers to predict the behavior of the system through simulation [59]. Analysis models are required to support engineering design decisions and enable engineering designers to evaluate the performance of a system using virtual representations. In the following sections, the information models and DL representations are presented for engineering analysis models. In Section 4.6.1, the generic information representation is presented, followed by an extension to the base vocabulary for representing specialized classes of analysis models in Section 4.6.2.

4.6.1 Generic Engineering Analysis Model Information Representation

The vocabulary defined in Table 4-2 and Table 4-3 is used with the DL constructs (Table 3-2) for specifying the information associated with engineering analysis models. Several examples are presented to demonstrate the use of the vocabulary for specifying engineering analysis models. The AnalysisModel concept captures the declarative information associated with engineering analysis models. This is both a strength and weakness of the representation. On the one hand, the declarative representation enables engineering analysts to implement and code analysis model knowledge in a programming

language or software application that is independent of the vocabulary. Thus, analysis experts are able to decouple the executable code from the declarative knowledge that is represented. However, because the declarative and executable representations are developed independently the analysis experts must ensure "consistency" between the representations. In the context of this research, a declarative representation of analysis model information is proposed that is not tied to a particular solution tool or method. The graphical representation of the information associated with the AnalysisModel concept is presented in Figure 4-23.

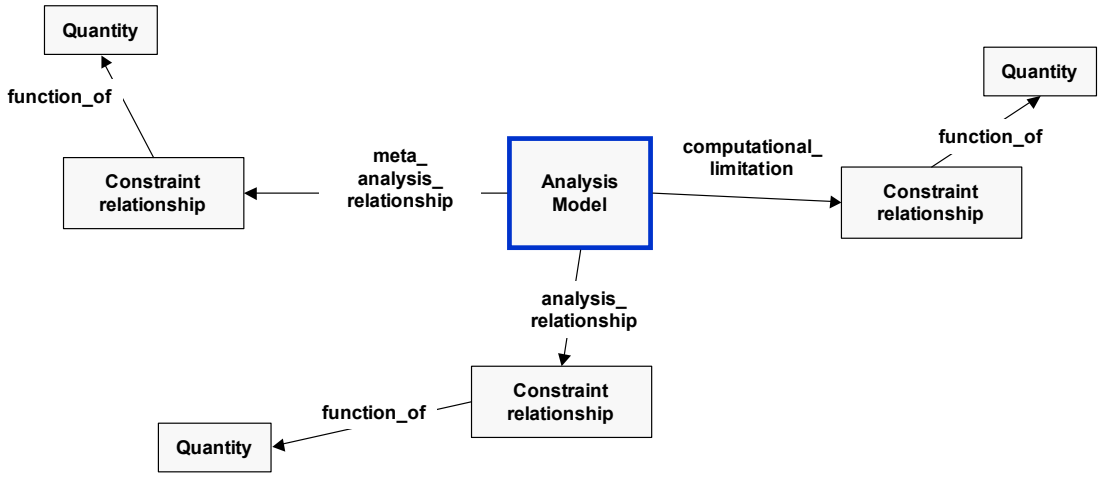


Figure 4-23: Graphical representation of AnalysisModel concept

As illustrated in Figure 4-23, representing executable code and analysis models is beyond the scope of the AnalysisModel concept. The AnalysisModel concept provides a means to capture analysis relationships, meta-level relationships (e.g., assumptions and limitations), and computational limitations as ConstraintRelationship concept. The information representation for engineering analysis models is comprised of three properties that specialize the constraint relationships concepts. The information associated with engineering analysis models is represented using DL as:

```
Class(AnalysisModel
and (
  (∃ analysisrelationship.ConstraintRelationship)
  (∀ analysisrelationship.ConstraintRelationship)
  (∀ computational_limitation.ConstraintRelationship)
  (∀ analysis_meta_relationship.ConstraintRelationship)))
```

The AnalysisModel concept is described in prose as the following:

"An analysis model has at least one analytical relationship, where all analytical relationships are constraint relationships ***and*** *all meta-relationships are constraint* ***and*** *all computational limitations are constraint relationships"*

The design decision information model is updated to reflect the addition of the AnalysisModel concept to the information base in Figure 4-24. The AnalysisModel concept specification is an independent of the compromise decision support model and variant previously defined in the information model. As illustrated in Figure 4-26, the AnalysisModel concept is not subsumed by the CDSP concepts or its variants. The AnalysisModel concept is defined using similar property specifications as the cDSP concept(s). However, because the concepts are defined using necessary and sufficient conditions, the AnalysisModel concept is not subsumed by the CDSP concepts.

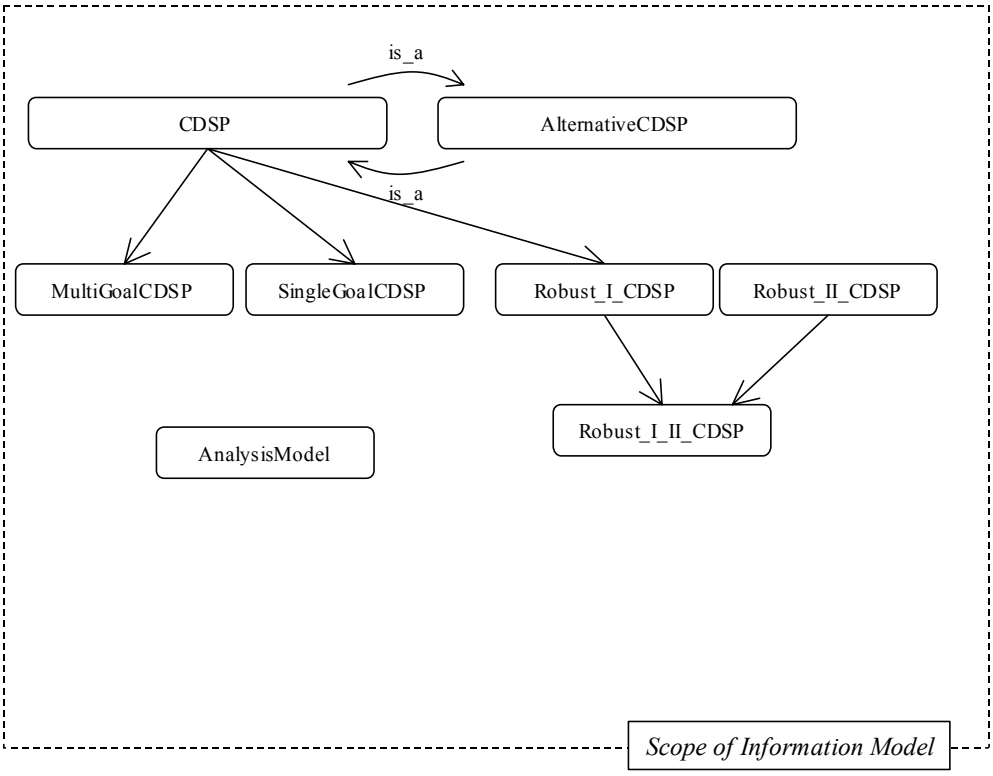


Figure 4-24: Decision construct information model – State 8

The AnalysisModel concept enables disciplinary analysis models to be integrated into engineering design decisions with a minimal ontological commitment [60]. In other words, while the executable information associated with disciplinary analysis models is not captured in the AnalysisModel concept, sufficient information is represented to enable disciplinary analysis models to be integrated in engineering decision constructs. The AnalysisModel concept definition developed in this research provides an "interface" between disciplinary analysis models and the cDSP. The AnalysisModel concept enables the quantities associated with engineering analysis models to be explicitly captured and organized in the context of constraint relationships.

The AnalysisModel concept acts as a wrapper for capturing the constraint relationships and meta-level relationships associated with a particular analysis or

simulation representation. The AnalysisModel concept is defined in a manner to facilitate reuse and extensibility. For example, the AnalysisModel concept is used in this dissertation for describing equations-based relationship and meta-level relationships.

The information associated with engineering analysis models is described using the base vocabulary established in Table 4-2 and Table 4-3. The base vocabulary enables many different types of analysis models to be specified. However, it does not provide a means for defining specialized analysis models, namely equations-based or computational analysis models. Gross and co-authors [59] establish a taxonomy and detailed classification of different types of engineering analysis models. However in this research, the scope is limited to equations-based and computational analysis models. In the next section, extensions to the base vocabulary are discussed and validated to ensure a richer representation of engineering analysis models can be specified.

4.6.2 Information Representations for Specialized Classes of Analysis Models

In the previous section, the generic AnalysisModel concept definition was specified. The base vocabulary is used to specify the declarative information structure of engineering analysis models. A similar problem arises with engineering analysis models as with design decisions - *How can the vocabulary be extended to enable modeling of concepts that fall outside of the current language and what are the implications of extending the base vocabulary?* As discussed in Chapter 3 and demonstrated in Section 4.5.4, the base vocabulary can be extended to represent information concepts not initially considered with minimal propagation and impact on existing concept definitions.

Two common types of engineering analysis model include equation-based model and computational analysis models. Equation-based analysis models rely on an equation that results in a closed-form solution for modeling a particular behavior of interest. The equation may be derived based on first principles, empirical data, or heuristics. Computational analysis models are solved using numerical techniques where closed-form solutions do not exist and result in approximations of a particular behavior of interest. In this context, a common type of computational analysis models is finite element analysis (FEA) models.

The engineering analysis models discussed in this research, including equation-based and computational, are evaluated using computer-processible representations. For example, equation-based models may be implemented using constraint solvers or as functions in traditional programming techniques. Similarly, computational models may be represented as parameterized simulation models in which input and output to the simulation models are represented as interface parameters. An example of parameterized simulation models include the "wrapper technology" used in Phoenix Integration's ModelCenter [2]. While the distinction between equation-based and computational analysis models is clear in this research, the actual distinction is difficult. For example, a finite element model is defined as a computational analysis model. However, the constitutive equations and relationships for elements in the model may be considered equation-based model. For the purposes of this research, equations-based models refer to those models in which the behavior / response of interest can be obtained by evaluating a governing equation. Additionally, computational models are often used interchangeably

with parameterized simulation models in which the governing equation is not explicitly known.

To specify the information associated with each of the specialization of the engineering analysis model, two additional concepts are added to the vocabulary. The concepts are asserted to be subclasses of the Relationship concept (see Table 4-8).

Table 4-8: Additional concepts specified in the decision-centric language

Concept	Definition
EquationRelationship	A specialization of the Relationship concept that captures an equation-based representation for a ConstraintRelationship
ParameterizedSimulationModel	A specialization of the Relationship concept that captures a computational relationship for a ConstraintRelationship concept

The ConstraintRelationship concept remains a generic container for capturing the associations between quantities and is used for defining engineering analysis model. The ConstraintRelationship concept is defined in Section 4.4. Specializations of the AnalysisModel concept are defined by creating specialization of the ConstraintRelationship concept. The information model for the ConstraintRelationship concept, originally presented in Figure 4-10, is specialized to capture the equation-based and computational relationships.

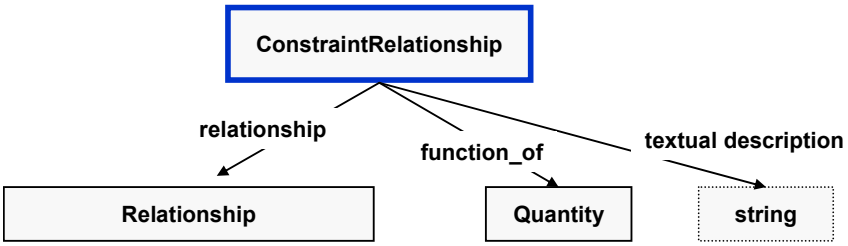
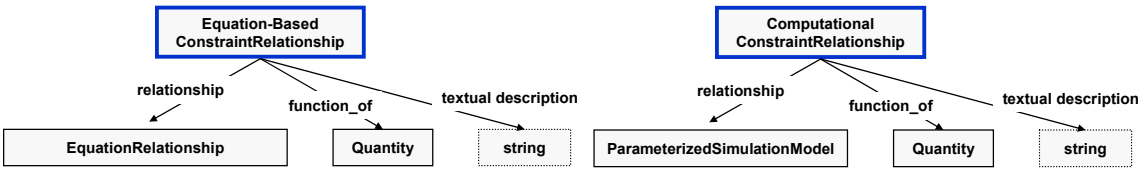


Figure 4-10: Graphical representation of ConstraintRelationship concept (repeated)

The specialized ConstraintRelationship concepts are defined by using the base concepts added to the vocabulary in Table 4-8. The graphical representation of the Equation-BasedConstraintRelationship and ComputationalConstraintRelationship are given in Figure 4-25.



(a) Equation-based ConstraintRelationship **(b) Computational ConstraintRelationship**

Figure 4-25: Graphical representation of specializations of ConstraintRelationship concepts

The resulting hierarchy for the ConstraintRelationship concepts is given in Figure 4-26.

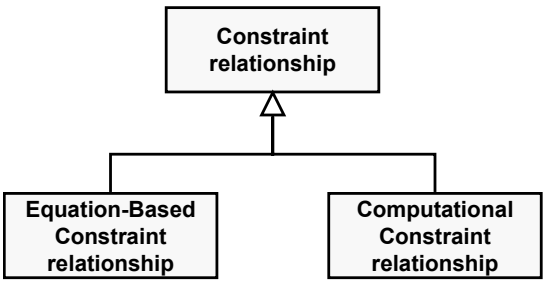


Figure 4-26: ConstraintRelationship concept hierarchy

As illustrated in Figure 4-27, two child classes are specified based on the concept specifications. The concept definition of the EquationBasedConstraintRelationship is given in DL as:

```
Class(EquationBasedConstraintRelationship complete
and(
(∃ function_of.Quantity)
(∀function_of.Quantity)
(∃ relationship.EquationRelationship)
(∀ relationship.EquationRelationship)
(= relationship 1)
(= textualdescription 1)))
```

The EquationBasedConstraintRelationship concept is given in English as:

"An equation-based constraint relationship has exactly one relationship that is an EquationRelationship, where all relationships are EquationRelationships and exactly one textual description."

Similarly, the ComputationalConstraintRelationship is given in DL as:

```
Class(ComputationalConstraintRelationship complete
and(
(∃ function_of.Quantity)
(∀function_of.Quantity)
(∃ relationship. ParameterizedSimulationModel)
(∀ relationship. ParameterizedSimulationModel)
(= relationship 1)
(= textualdescription 1)))
```

The ComputationalConstraintRelationship concept is given in English as:

"A computational constraint relationship has exactly one relationship that is a ParameterizedSimulationModel, where all relationships are ParameterizedSimulationModels and exactly one textual description."

As a result of adding concepts to the base vocabulary, the Relationship concept definition is changed from an atomic concept to a defined concept. To ensure the hierarchy for the constraint relationship concepts is consistent, additional concept assertions must be stated. The first assertion states that the ParameterizedSimulationModel and EquationRelationship concepts are disjoint concepts. In other words, the ParameterizedSimulationModel and EquationRelationship concept are mutually exclusive. The second concept assertion is a closure axiom that states the Relationship concept is either a ParameterizedSimulationModel or an EquationRelationship. The closure assertion states the ParameterizedSimulationModel and EquationRelationship concept are collectively exhaustive. The definition for the Relationship concept is given as:

```
Class(Relationship complete
and(
(¬ParameterizedSimulationModel EquationRelationship)
(ParameterizedSimulationModel ∪ EquationRelationship)))
```

The concept definitions for the equation-based analysis model and computational analysis model are derived directly from the specializations of the constraint relationship concepts. The equations-based engineering analysis model and computational analysis model graphical information models are represented in Figure 4-27 and Figure 4-28.

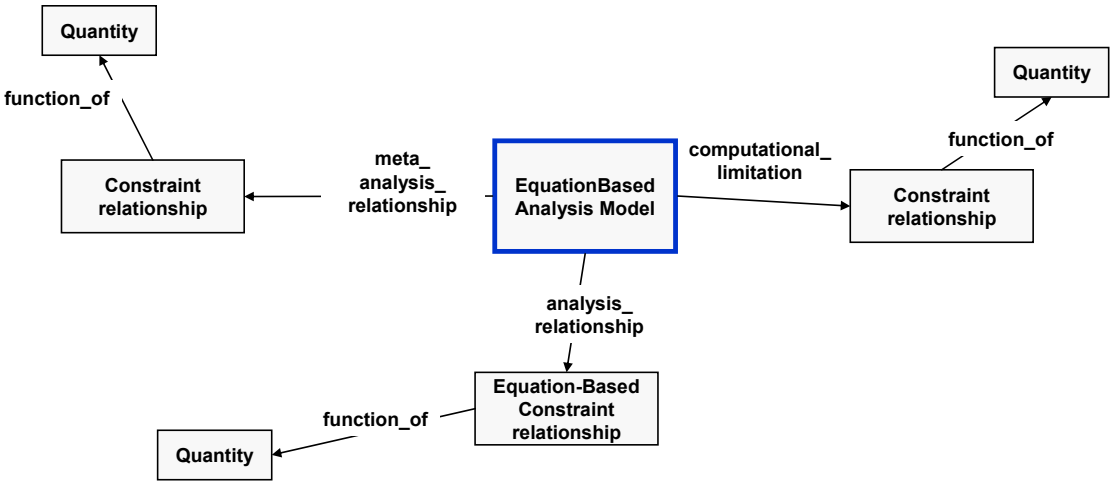


Figure 4-27: Graphical representation of EquationBasedAnalysisModel concept

The EquationBasedAnalysisModel concept is defined in DL as:

```
Class(EquationBasedAnalysisModel
and (
  (∃ analysisrelationship.EquationBasedConstraintRelationship)
  (∀ analysisrelationship.EquationBasedConstraintRelationship)
  (∀ computational_limitation.ConstraintRelationship)
  (∀ analysis_meta_relationship.ConstraintRelationship)))
```

The EquationBasedAnalysisModel concept is described in prose as the following:

"An equation-based analysis model has at least one analysis relationship, where all analysis relationships are equation-based constraint relationships ***and*** *all meta-relationships are constraint relationships* ***and*** *all computational limitations are constraint relationships."*

Similarly, the ComputationalAnalysisModel concept is illustrated graphically in Figure 4-28.

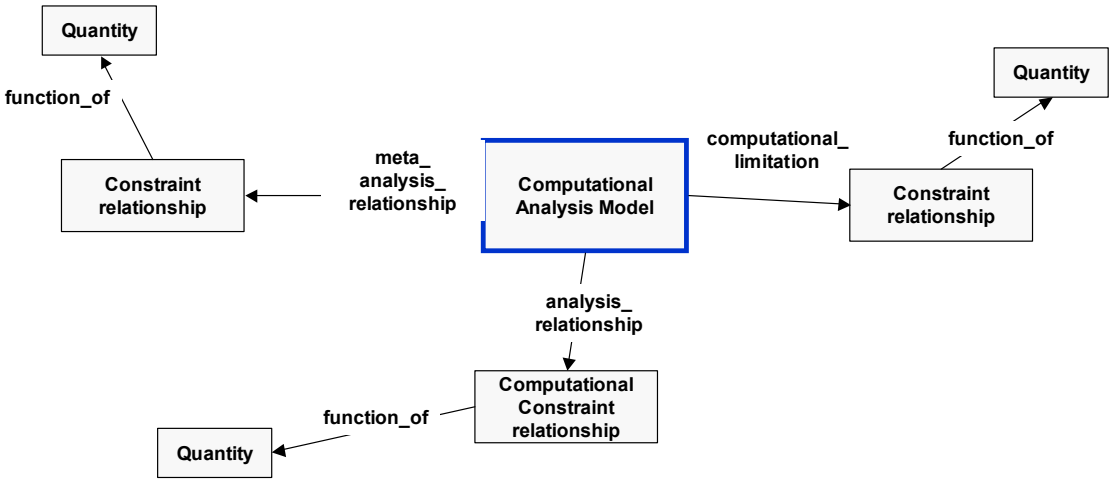


Figure 4-28: Graphical representation of ComputationalAnalysisModel concept

The DL concept definition for the computational analysis model is given as:

```
Class(ComputationalAnalysisModel
and (
  (∃ analysisrelationship.ComputationalConstraintRelationship)
  (∀ analysisrelationship. ComputationalConstraintRelationship)
  (∀ computational_limitation.ConstraintRelationship)
  (∀ analysis_meta_relationship.ConstraintRelationship)))
```

The ComputationalAnalysisModel concept is described in prose as the following:

"A computational analysis model has at least one analytical relationship, where all analytical relationships are computational constraint relationships ***and*** *all meta-relationships are constraint* ***and*** *all computational limitations are constraint relationships."*

Figure 4-29 illustrates the modifications to the design decision information model through the specification of the computational and equation-based analysis models.

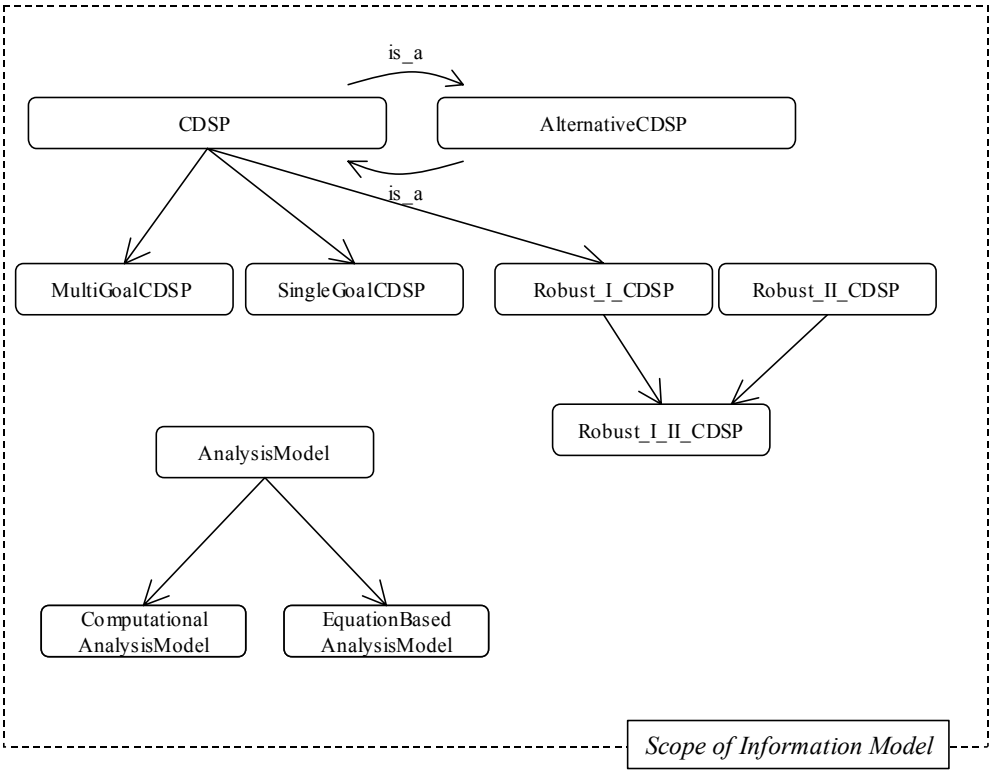


Figure 4-29: Decision construct information model – State 9

The ComputationalAnalysisModel and EquationBasedAnalysisModel concepts are subsumed by the AnalysisModel concept. As previously discussed, the specialized concepts captured increasingly restrictive definitions. For example, the ComputationalAnalysisModel must have a constraint relationship that is *linked* to a Parameterized simulation model. The AnalysisModel concepts and subclass concepts are not subsumed by the CDSP (and associated sub-concepts). As illustrated in Figure 4-29, the AnalysisModel and CDSP concept network are related through subclass/superclass associations. The concepts are related because they use similar atomic concepts and properties. For example, both the analysis model and cDSP concepts have the Quantity concept in common. Thus, the concepts are inter-related through the *common associations.* A detailed discussion and the importance of this characteristic is presented

in Section 4.7. However, because the concept definitions are specified using necessary and sufficient conditions, the AnalysisModel concept is not subsumed by the CDSP concepts. A brief examples and discussion on necessary and sufficient conditions is provided in Table 4-9.

Table 4-9: Summary of necessary and sufficient conditions

Condition	Discussion and example
Necessary	A necessary condition is one that must be satisfied for the result to occur. Necessary conditions are discussed in the context of if and only if (iff) statements. For example: A is necessary for B iff B can't occur without A. In other words, if B occurs then A also occurs. In terms of DL representations all B concepts are A concepts
Sufficient	A sufficient condition must be satisfied for the result to occur. Similarly, sufficient conditions are often discussed using iff statements. For example, A is a sufficient condition of B iff A guarantees that B will exist. For example, if A occurs then B will also occur. In terms of DL all A concepts are Bs.
Necessary & Sufficient	Necessary and sufficient conditions are increasingly restrictive by combining the Necessary and the Sufficient conditionality.

The vocabulary is used for building concept definitions and specifying the information structure associated with engineering analysis models. As previously stated, the AnalysisModel concept and sub-concepts provide a wrapper technology for capturing the constraint relationships and meta-level relationships associated with engineering analysis models. The AnalysisModel concept is defined in a manner to facilitate reuse and extensibility independent of a particular simulation code or application.

Up to this point, engineering design decisions and analysis models have been discussed and formal definitions have been specified independently. For example, the

definitions of the multi-objective engineering design decisions have been specified with minima discussion of engineering analysis models and the integration of each concept into a unified decision construct. In the following section, the concept are integrated into a unified decision construct. A critical review and discussion of the value, shortcomings, and implications of the integrated concepts is then presented.

4.7 Integration and Representation of cDSP and Analysis Model Concepts

In the previous discussion, the CDSP and AnalysisModel concepts (and closely related variants) are discussed individually. In other words, the information associated with each of the constructs is not presented in an integrated manner. The reason for discussing the concept in this manner is twofold. First, the cDSP and engineering analysis model concepts are presented individually to highlight the decoupled nature of disciplinary analysis models and design decisions. By presenting each of the concepts separately, the notion of analysis model reuse across design decisions is emphasized and the ability for disciplinary analysis models to be integrated into a unified decision construct is illustrated. This approach parallels the decision formulation method presented in Figure 4-2, Figure 4-3, and Figure 4-7. Second, the network of relationships and associativities between information in the analysis models and design decision becomes is complex. The concept definitions and DL specifications for the analysis models and cDSP concepts remains the same. Additionally, the graphical information representation of the individual concepts does not change. However, the relationships between the concepts are reflected in the links between analysis quantities and decision quantities (see Figure 4-30).

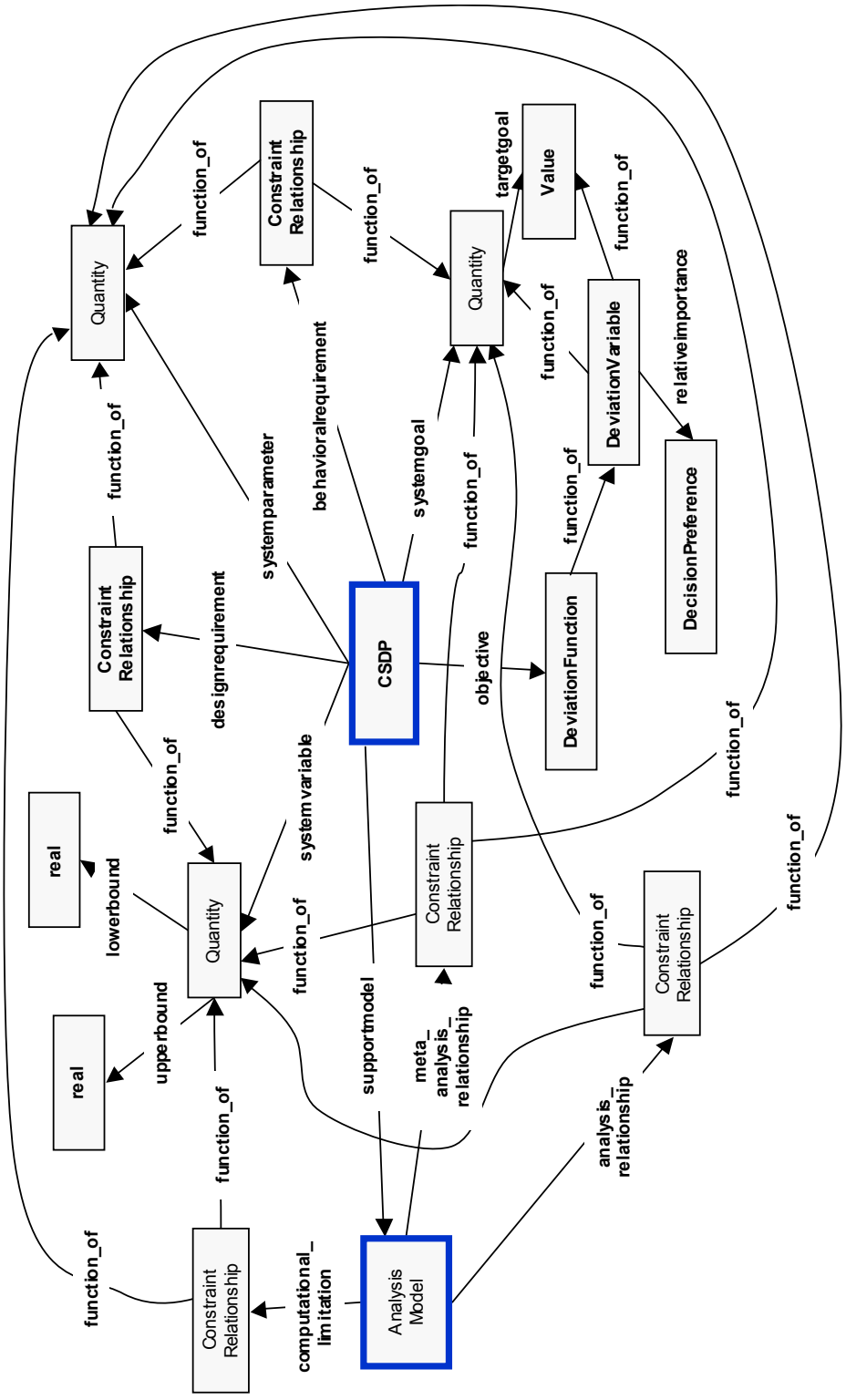


Figure 4-30: Graphical representation of integrated cDSP and AnalysisModel concepts

The relationships between the AnalysisModel concept and cDSP concept are represented as directed arrows. As expected, the information network and relationships between the concept associated with the AnalysisModel and the CDSP become increasingly complex. The individual concepts are related through the Quantity concepts. For example, the AnalysisModel and CDSP concepts are related to the Quantity concept through the function_of property. Thus, the common information shared between AnalysisModel concepts and CDSP concepts are captured in the Quantity concept. The "digital interface" between the AnalysisModel and CDSP concepts is illustrated in Figure 4-31.

The shaded boxes in Figure 4-31 represent the digital interface between the cDSP concepts and the analysis model concepts. The development of digital interfaces has been the subject of research in engineering decision making for some time [48; 166]. Unfortunately, researchers have failed to settle on general accepted definition of digital interfaces for engineering decision making. Aside from inability to settle on an accepted definition, previous researchers have not adequately addressed information representation and exchange. For example, the foundational research from the database design, knowledge representation, and engineering information management communities has not been leveraged in the discussion and development of digital interfaces for engineering design decisions. Thus, existing digital interfaces for engineering design decision have only addressed information exchange from a philosophical standpoint. While this interpretation of a digital interface is not entirely incorrect, it does not adequately address the needs of modern product development and decision making in which a significant portion of product information is created and represented in digital format.

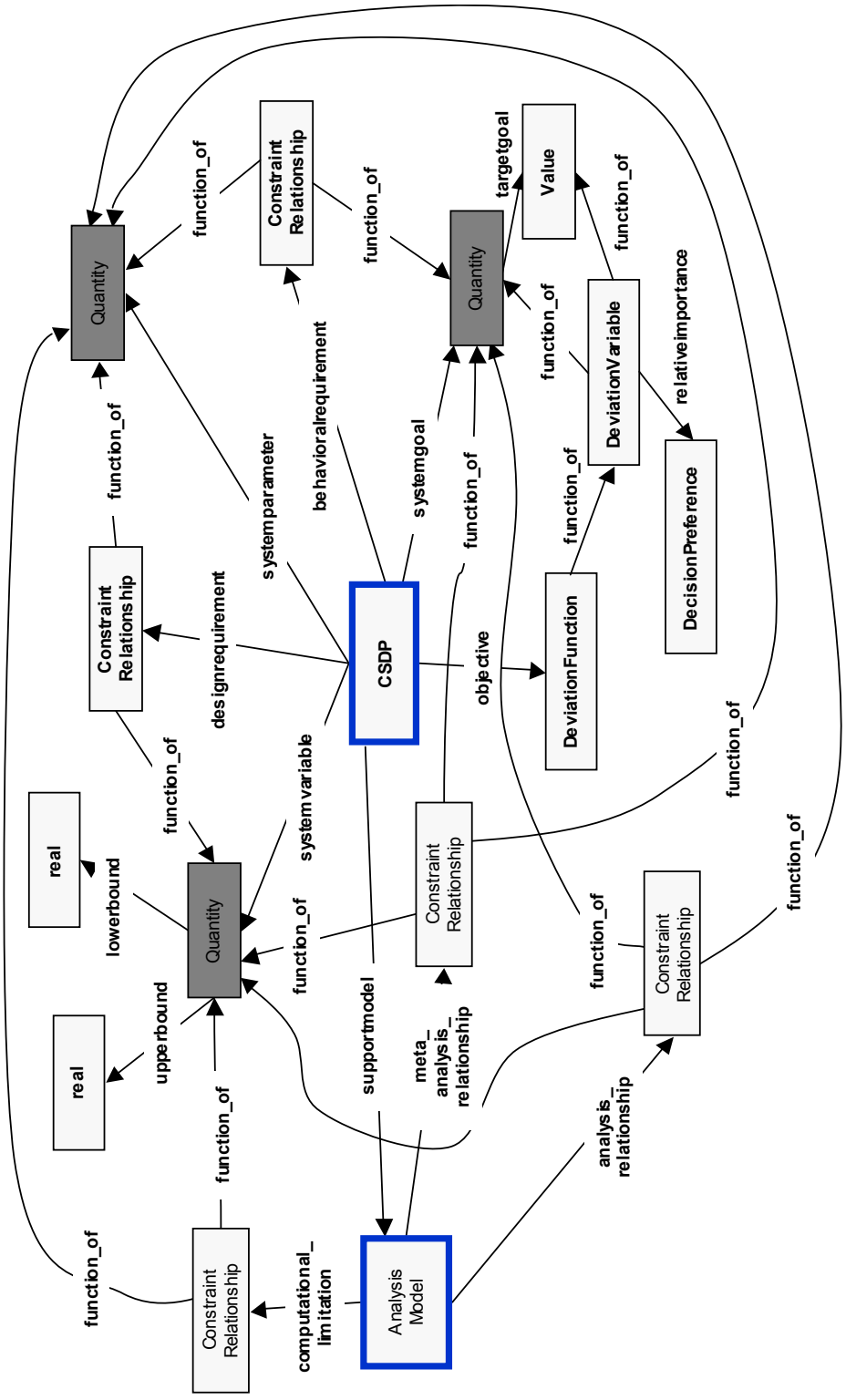


Figure 4-31: Digital interface between cDSP and AnalysisModel concepts

The digital interface between disciplinary analysis models and engineering design decisions is realized through the formal representation of the Quantity concept. For example, the quantities associated with engineering analysis models are *published,* thus provided a prescribed set of information that is required to execute an analysis models for a particular design decision. Similarly, the information required to complete a design decision can be specified, thus enabling disciplinary analysts to plug-and-play various analysis model into design decisions. The integrated decision concept presented in Figure 4-31 is a generic representation of the information in CDSP and AnalysisModel concepts. Specific discussions and information representations for particular design decisions and analysis models are presented in Chapters 5 and 6.

4.8 Discussion and Critical Assessment of Formal Language

Several components are presented in this chapter for developing formal representations of decision-related information including: (1) a systematic method for formulating engineering design decision, (2) a vocabulary of the concepts and properties associated with multi-objective design decisions that constitute the formal language, (3) a graphical representation and notation of the information models for multi-objective design decisions and analysis models, and (4) DL concept definitions based on the vocabulary that provide a computer interpretable representation. The components, collectively, provide a means for unambiguously representing and exchanging decision-related information between multiple engineering design disciplines. The components presented in this chapter closely parallel the contributions in this research.

As stated in Chapter 3, there are a myriad of information models that have been developed for addressing various aspects of engineering design. Thus, the question arises,

How can the correctness and completeness of the information model be determined? The goal in this section is to establish the formal language and information model proposed in this research addresses a gap not currently addressed by other information modeling efforts. Of particular interest in this discussion is how to verify the information model is correct. In this section, the components of the formal language are critically evaluated in the context of the information modeling requirements presented in Sections 3.5 and 4.1 and criteria put forth by Gruber [60] (see Table 4-10). The design criteria put forth by Gruber is modified slightly for assessing the formal language.

Table 4-10: General criteria for assessing ontologies (adapted from [60])

1. **Clarity:** A formal language should effectively communicate the intended meaning of defined terms. Definitions should be *objective.* All definitions should be documented with natural language.
2. **Coherence**: A formal language should be coherent: that is, it should sanction inferences that are consistent with the definitions.
3. **Extendibility**: A formal language should be realized to anticipate the uses of the shared vocabulary. It should offer a conceptual foundation for a range of anticipated tasks, and the representation should be crafted so that one can extend and specialize the ontology monotonically. In other words, one should be able to define new terms for special uses based on the existing vocabulary, in a way that does not require the revision of the existing definitions.
4. **Minimal encoding bias**: The conceptualization should be specified at the knowledge level without depending on a particular symbol-level encoding. The tendency toward a particular modeling or programming language should be minimized.
5. **Minimal ontological commitment**: The formal language should impose minimal ontological commitment sufficient to support the intended knowledge sharing activities. Since commitment to a language is based on consistent use of vocabulary,

ontological commitment can be minimized by specifying the weakest theory (allowing the most models) and defining only those terms that are essential to the communication of knowledge consistent with that theory.

A summary of the evaluation is presented in Table 4-11, followed by a detailed discussion of the limitations and strength of each of the components individually as well as collectively.

Table 4-11: Assessment of information model versus engineering requirements

R1. Extensibility	✓	The vocabulary and DL representation have been illustrated to be extensible by developed complex concepts from the initial set of concepts and properties. In addition, new concept and properties are added and/or modified to the base vocabulary to enable robust design decision and specialized analysis models to be modeled. The effort required to extend the vocabulary is minimal and the changes do not propagate to existing concept definitions. The evolution of the decision information model illustrate how new concepts, at the sub-class and super-class level can be specified.
R2. Consistency	✓	The consistency of the information base is illustrated by dynamically organizing the hierarchy of engineering decisions. This is particularly important as new concepts are added. The relationships between concepts are automatically determined. Additionally, equivalent concepts and unsatisfiable concepts are determined based on concept definitions and DL reasoning
R3. Information organization	✓	The decision making concepts defined were automatically organized into a hierarchy based on concept definition. The class was updated independent of the order in which concepts were defined.

Table 4-11: Assessment of information model versus engineering requirements (continued)

R4. Systematically capture	O	The information model is developed based on a systematic method. The method provides the basic scaffolding for formulating design decisions. The original vocabulary is developed in the context of

information from multiple perspectives		the systematic method. However, the method originally developed does not provide support for alternative decision formulations. For example, the method reflects the needs of formulating traditional cDSPs, but does not reflect robust cDSP.
R5. Computer interpretable	✓	The information representation of engineering design decision is computer interpretable. Description Logic and OWL are used to capturing the knowledge associated with design decisions.
R6. Analysis model integration	✓	The vocabulary minimizes the ontological commitment and enables disciplinary analysis models to be integrated into design decisions based on a digital interface. The digital interface is based on the aggregation and commonality of the Quantity concept.
R7. Reuse and Retrieval	O	Reuse of knowledge is demonstrated by specializing and creating new concepts based on existing concepts. Retrieval of decision information is not explicitly illustrated
R8. Predefined Vocabulary and Structure	✓	The information model relies on a basic set of predefined concepts and property definition. The information model does not enforce a database "schema" but rather provide a means for defining complex concepts using the vocabulary in a flexible manner
R9. Limitations and Assumption of Analysis Models	✓	The limitations and assumptions of the analysis models are captured for use in the design decisions. The analysis model representation explicitly captured the limitations of the analysis models used in design decision.
R10. Understood by Designer	✓	Because the vocabulary relies on an unambiguous definition, it is likely the information model will be understood by designers and analyst
Key: ✓: Fully met; O: Partially met; ×: Not met		

Systematic Method. The systematic method provides a structured means for explicitly capturing the information associated with engineering design decisions. The systematic method is based on current literature for modeling multi-objective design decisions as compromise decision support problems. The cDSP is a well-accepted formulation for modeling decision commonly encountered in engineering design. The method provides a basis for capturing and integrating decision-related information from

multiple perspectives and consists of seven phases. Mathematical-based representations of engineering decision information are developed in Section 4.2 based on the phases and steps for decision formulation. These mathematical representations provide the first step in bridging the gap between the mathematically formulated optimization problem and the information-based decision problem. For example, matrix and array notation are used to capture the information associated with design variables, system goals, analysis models, design requirement, and the cDSP construct. The mathematical representations are used for developing the vocabulary, graphical information models, and DL implementation.

The method is decomposed into two closely-related sub-methodologies. The first sub-method is focused on capturing the "core" information associated with engineering decisions including the representation of design variables, design parameters, system goals, design and analysis constraints, and decision preferences. The first sub-method consists of five phases and provide a means for structuring the information associated with a design decision without committing to specific analysis model. The second sub-method comprises a single phase consisting of four steps. The second sub-method provide as structured basis for explicitly capturing analysis-related information. The second sub-method is targeted towards disciplinary analysis experts by providing a mean for publishing information about complex engineering analysis models that enable seamless integration with engineering design decisions. This decomposition is chosen to enable disciplinary analysts to systematically capture and "publish" analysis models independent of design decisions.

The decomposition of the method occurs at *natural breaking point.* The first sub-methodologies encapsulate the phases and steps that are associated with multi-

disciplinary decision making. The information captured in the first method provides a means for declaring the information of interest for a design decision. However, sufficient freedom remains that enables disciplinary experts from multiple domains to develop and implement different analysis models. Conversely, the interface between the first and second sub-method provides a means for developing analysis models somewhat independently of the decisions in which they may be used. The advantages of this decomposition lie in the fact that the multi-disciplinary decision formulation and the implementation of disciplinary analysis models are decoupled. This implies that not all the information associated with a complex multi-disciplinary design decision must be exchanged between all disciplines. Only those disciplines that must share or exchange information must communicate, thus simplifying and reducing the shared information between disciplinary analysis models while retaining the complex inter-disciplinary information exchange at the decision level.

The systematic method provides tremendous advantage and structure for modeling engineering design problems as multi-objective optimization problem. However, there are a few shortcoming associated with the method. First, the method is currently limited to modeling and structuring the declarative information associated with the cDSP decision construct and analysis support models. While the cDSP is a valid formulation of engineering design decision, it is not the only formulation. There are several other mathematical formulations that provide a basis for representing multi-objective engineering design decisions. The current method, as demonstrated in this research, is limited to cDSP formulations. In fact, several steps in the systematic method must be

modified accurately represent the information associated with robust decision model presented in Section 4.5.4 (see Figure 4-32).

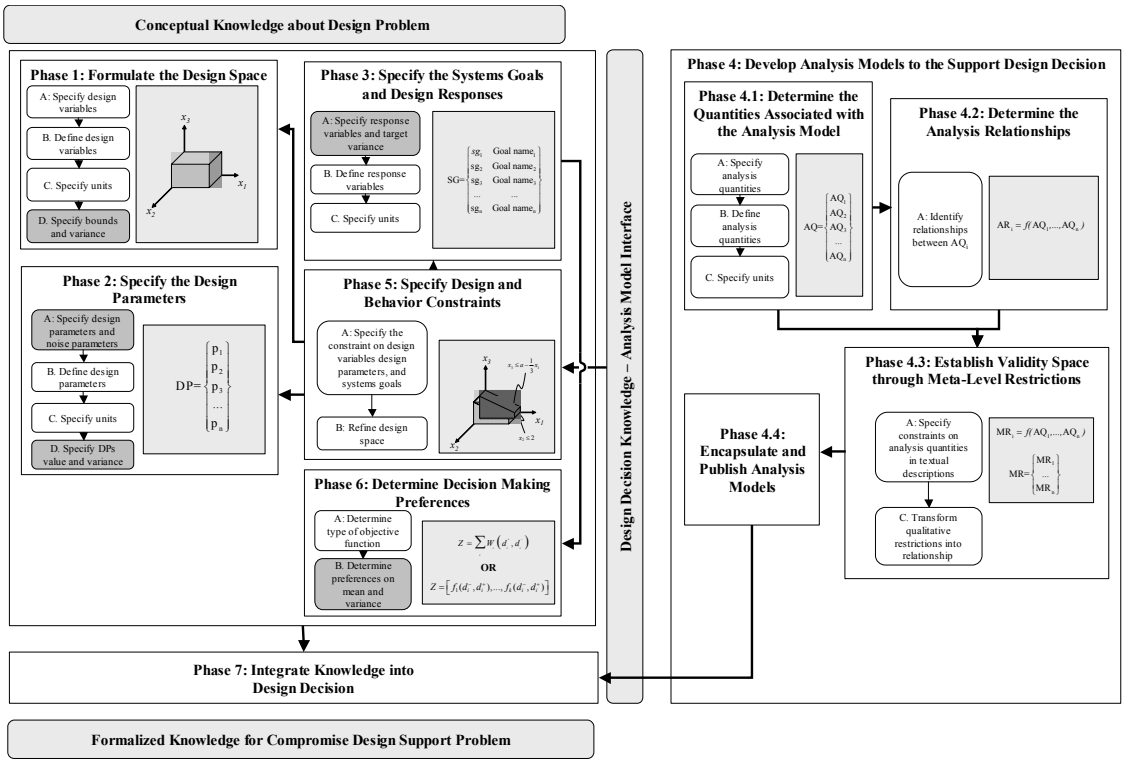


Figure 4-32: Systematic method for formulating robust design decisions

The dark gray boxed in Figure 4-33 indicate the five steps that are modified to reflect the additional information required for modeling Type I and Type II robust design decisions. The method is focused on capturing the information associated with the traditional cDSP construct and thus provides a systematic approach for capturing the relevant information. However, the method is flexible and can be modified to reflect changes in the underlying decision model. One important caveat is that consistency between the systematic method (i.e., the process model) and the information model of the cDSP must be ensured. Additional research is needed to identify integrated process and information modeling techniques for engineering design decisions.

The second limitation is related to the disconnect between the declarative information representation and the executable implementation. This research is focused on representing the declarative information associated with engineering design decision and therefore is not tied to a particular solution method or tool. This characteristic is both a strength and a limitation. It is a strength because decisions can be modeled in a manner that is independent of a particular solution technique or programming language. For example the declarative information associated with a specific design decision will remain the same whether the decision is solved using an exhaustive search or gradient based approach or whether the decision is implemented using MATLAB or C-programming. This enable designers to represent the decision-related information in a means that they are accustomed, not in a particular programming language. With that said, because the declarative representation is not tied to a particular language, significant additional effort is required to represent the information in a form that can be solved. Additionally, the procedural decision representation must be consistent with the declarative representation – a task that becomes increasingly difficult as the decision become increasingly complex. For the scope of this research, the bridge between declarative and procedural representation is important to address, but it determined to be future work.

Design Decision Vocabulary. A vocabulary is developed based on the mathematical information representations identified from the systematic method. The vocabulary consists of a predetermined set of concepts and properties used for describing the information associated with multi-objective engineering design decisions. The semantics of the vocabulary are established by developing definitions of each of the basic concepts

and properties. In addition, the domain and range of properties are specified, thus restricting the properties to a predetermined set of concepts. The vocabulary provides several advantages, some of which are indirectly discussed in the systematic method section, for modeling the information associated with engineering decisions. First, the vocabulary is formal. In this context, formal refers to explicitly capturing and representing the entities and relationship for a domain of discourse in a formal language. In this research, the formal language for representing engineering decisions is based on a description logic language and the proposed vocabulary. Additionally, another important feature is the vocabulary and language are both human-interpretable and computer-processible. As illustrated in Table 4-2 and Table 4-3, the vocabulary is expressed in natural language that can be understood by engineering decision makers and can be processed in a computational manner. This definition is similar to that developed by Gruber for the development of ontologies. The vocabulary is developed such that disciplinary analyst are able to capture the quantities and constraint associated with analysis models, without knowing how the analysis models will be coupled to other discipline and independent of specific design decisions. Additionally, the vocabulary enables decision makers to model complex engineering decisions using a basic set of concepts and properties. The vocabulary also enables decision makers to integrate disciplinary analysis models without knowing what "goes on under the hood" of the models.

The design decision vocabulary is objective. The vocabulary is rooted in mathematically-based representations for engineering design variables, design parameters, system goals, and constraints. Based on these representations a set of

predetermined natural language definitions are established. Additionally, the concepts and properties that constitute the vocabulary are not tied to a particular modeling domain or engineering disciplines and thus can be used for representing complex concepts across multiple domains. However, the vocabulary is limited to modeling multi-objective design decision that are represented as cDSP. The vocabulary minimizes the encoding bias. First, the vocabulary is specified in a declarative manner, independent of a particular programming language. The vocabulary is not tied to a particular application and relies on generic data type property specification, not application specific data types. For example, Additionally, when data type properties are used, the data types are specified in a generic manner. The vocabulary is extensible. The terminology in the vocabulary provides a means for *building* complex concept definitions, thus providing a means for developing new terms using existing concept definitions while not effecting the existing definitions. This is illustrated through the specialization multi-goal and single goal cDSP constructs and through the addition of robust decision concepts. The vocabulary requires minimal ontological commitment. Gruber discusses ontological commitment in terms of the generality of the ontology (in our case the vocabulary). The vocabulary is developed and only the essential terms are defined. For example, after several iterations it was determined that the Quantity concept was a primary means for communicating between disciplines. Thus, disciplinary engineers and designer need only to agree on an established set of Quantities in order to communicate between disciplines and integrate information in multi-disciplinary design decisions.

Graphical Information Model. The graphical information models for representing the network of concepts and properties associated with design decisions. The graphical

representations of the information models provide a means for visualizing the information and relationships associated with engineering design decisions. The representations become increasingly complex and the number of relationships becomes more dense as does the complexity of the decision concepts. For example, the information models presented at the beginning of the chapter consist of only a few concepts and properties (i.e., ConstraintRelationship and Quantity concepts), while the information representations at the end of the chapter are a complicated web of concepts and properties (i.e., the integrated representation of cDSP and analysis model concepts). However, the complexity of the graphical representations does not outweigh the importance and value of developing the graphical information representations. The graphical representation enable the "linkages" between the quantities (i.e., design variables, parameters, and system goals) to be visualized and the relationships between these parameters and design constraints and analysis models. For example, a designer is able to quickly determine what information is required or what additional information must be generated to execute a decision. The graphical representation serves as a prescriptive template of what information must be instantiated. Additionally, the graphical representation enables a designer to determine what analysis models are driven a design decision and how the analysis models are coupled. The graphical representations enable the linkages of disciplinary analysis models to be visualized through the Quantity concept. In this research the graphical representations are for the decision concepts are created manually. However, the task of creating and subsequently visualizing the information become arduous and the value decreases as the number of concepts and properties increases. Thus, tools are required to navigate, view, and modify the

information representations. However, the development of these tools is beyond the scope of this research and left as future work.

Description Logic Representation. The final component presented in this Chapter is the DL implementations of the decision-related concepts. As established in Chapter 3, description logic (DL) is chosen as the representational formalism for capturing the information associated with the cDSP construct. Unlike other knowledge representation approaches in which the concepts are created in an ad-hoc manner, DL uses a fixed set of primitives (i.e., concepts and properties) can be used to construct complex object descriptions. DL is used in conjunction with the base vocabulary for developing formal definitions of decision information and analysis models. Several information representations are developed (e.g., traditional cDSP, Robust I cDSP, Robust II cDSP, Robust I-II cDSP, and analysis model concept) using the vocabulary and DL constructs. Complex concepts are defined using a predetermined set of construct and vocabulary. As illustrated the DL representation provide a computer-interpretable representation of decision related knowledge that can be reasoned with.

Several examples (summarized in Table 4-1) are presented that illustrate the use of DL for representing and organizing the information models for capturing decision-related and analysis information. Standard reasoning services supported by DL are utilized to ensure the organization and consistency of cDSP information models.

Collectively, the systematic method, vocabulary, graphical representation, and DL implementation provide a means for enabling designers to explicitly capture the information associated with multi-objective design decisions in a manner that is human-interpretable and computer-processible. Most importantly, the language and

implementation developed in this research provide a digital interface for integrating and exchanging decision-related information in multi-disciplinary design problem.

As previously discussed, the development of digital interfaces for integrating and exchanging information in the context of engineering design decisions has been the focus of several research efforts. However, previous research efforts have not adequately addressed the need for formal information representations to enable the development of digital interfaces. In this research, we approach the development of digital interfaces from a computational perspective by establishing a formal language for representing engineering design decisions. We believe that this approach enables us to focus on the *digital* aspect and create representations that enable the exchange of product information. The underlying notion in this research is that *formal information model and vocabulary will provide a computational digital interface for exchanging information associated with engineering design decisions.* Hence, digital interfaces are emergent concepts that represent formal representation of decision-related information. With this approach were are able to address the philosophical level digital interface, by creating representation of design decisions that enable disciplinary information to be packaged and exchanged.

The limitations associated with the components presented in this chapter are centered on the fundamental disconnect between the declarative knowledge representation and procedural *implementations.* A summary is presented in this section.

- The information model is not linked to a particular solver or modeling environment – there is a gap between the declarative knowledge representation and executable code. Currently a manual translation process is required to execute the decisions

- The analysis information models is not linked to external analysis models – the ConstraintRelationship concept that captured analysis knowledge is an interface to external analysis code, but is not linked to a particular tool.
- The method for formulating design decision and the information model are not synchronized. The process and information models should be consistent and provide guidance to the decision maker as to what information is required for formulating a particular type of decision.

In the following section, the role of this chapter is discussed in terms of Verification and Validation.

4.9 Verification and Validation

In this chapter Theoretical Structural Validity (TSV), Empirical Structural Validity (ESV), and Empirical Performance Validity (EPV) are addressed (see Table 4-12).

Table 4-12: Validation and verification in Chapter 4

Theoretical Structural Validation
§4.1 – Several requirements are identified for capturing the information associated with engineering design decisions. These requirements are domains specific and provide the scope of the formal language developed in this research. These requirements in conjunction with those developed in §3.5 are used to evaluate and assess the formal language developed. The requirements provide a framework for verifying and validating the formal language. §4.2 – A systematic method is developed for capturing the information associated with the cDSP construct. The method captures the basis phases and steps followed in formulating a cDSP. The method provides a structured approach for explicitly capturing and organizing the information associated with decision formulation. The method is based on current cDSP literature and therefore the internal consistency of the method is supported §4.3 – A vocabulary, consisting of concepts and properties, is developed. The vocabulary is abstracted from the mathematical formulation of the cDSP and the systematic method. The vocabulary is internally consistent because it is abstracted from a decision construct that has been verified and validated.
Empirical Structural Validation
The appropriateness of the design decisions and analysis models is first established in Table 4-1. The examples provide a means for assessing the use of DL and the vocabulary for information modeling of engineering design decisions and analysis models. As summarized in Table 4-1, the examples provide a means for assessing the information model, vocabulary, and DL representation in the context of the requirement developed in §3.5 and §4.1. These include robustness, extensibility, information organization and consistency, modeling design decisions and analysis models

Table 4-12: Validation and verification in Chapter 4 (continued)

Empirical Performance Validation
§4.4 – 4.8 – The vocabulary is used for modeling a variety of engineering design decisions and analysis models. The vocabulary and DL representations are used for modeling the information associated with the example problems identified in Table 4-1. The strengths and limitations of the information models developed using the vocabulary and the DL implementation are critically assessed and discussed in the context of the engineering information modeling requirements. No quantitative results are obtained. However, a general discussion results about the implications associated with modeling the decision-related information using the vocabulary. Verification and validation are established by providing support through several examples.

As previously stated, Theoretical Structural Validation (TSV) refers to accepting the individual constructs constituting the method and accepting the internal consistency of the way the constructs are put together. Theoretical Structural Validation is carried out in this chapter using a systematic procedure consisting of 1) identifying the requirements and scope of information representation, 2) developing a systematic method based on existing literature, 3) establishing the core vocabulary for representing information associated with cDSP and analysis models, and 4) representing the information models in a computer-processible manner using DL. The constructs used in this chapter include the cDSP for modeling engineering design decisions and DL for developing computer-based information representations. The cDSP has been used in existing literature and design applications for modeling multi-objective design decisions. However, based on a critical review of existing literature, it wad identified that current research has not addressed the

development information-intensive nature of engineering design decisions. Additionally, a systematic method for modeling engineering design decision was not identified in current literature. In order to address the limitations, a formal language is established for modeling the information associated with design decisions. The formal language consists of a predetermined vocabulary of concepts and properties that are used in conjunction with DL as an information modeling formalism. The advantages of these construct individually have been shown in existing literature and Chapter 3, which provides confidence in the applicability of the integrated constructs. The integration and synergistic characteristics of these constructs results in several advantages and provide a means for explicitly capturing the information associated with design decision. Hence, we believe that a DL-based language for modeling the cDSP and associated analysis support models enable use to achieve TSV.

Empirical Structural Validation (ESV) refers to accepting the appropriateness of example problems used to verify the performance of the method. In this chapter, we use ten examples for validation the formal language, information model, and DL implementation of engineering design decisions. The examples differ in complexity, the information associated with the concept, and the DL construct used to specify the definition. The set of example problems include the traditional cDSP formulation, multiple and single goal formulations, and robust decision formulation. The examples are not tied to a particular domain or application, but rather are general models of multi-objective engineering design decisions. A summary of the example problems is presented in Table 4-1. The examples problems are chosen based on existing literature and current research associated with the cDSP. Each of these models has received significant

attention and has been applied in actual engineering scenarios. Thus, the example problems represent actual design decision problems. Additionally, the example problems are chosen based on the requirements and criteria established in Sections 3.5 and 4.1. The example problem must collectively address all of these requirements. As summarized in Table 4-1, the examples provide a means for assessing the extensibility, robustness, consistency, organization, and specific decision modeling requirements. Thus, we are confident that the example problems presented in this chapter are appropriate for verifying and validating the systematic method, information model, and DL implementation presented in this chapter.

Empirical performance validation (EPV) refers to accepting that the outcome of the method is useful with respect to the initial purpose for the chosen example problems and accepting that the achieved usefulness is linked to applying the method. It is shown in Sections 4.4 through 4.6 that vocabulary developed can be used in conjunction with DL for explicitly capturing the information associated with engineering design decisions. The formal language provides a human-interpretable and computer-processible means for capturing, exchanging, and integrating information in multi-objective engineering design decisions. The information representations presented in this chapter are dynamically organized based on the concept definitions, not explicit relationships established between concepts. This demonstrated the use of DL reasoners for maintaining the consistency and dynamically organizing decision-related information. Additionally, the extensibility of the language is demonstrated by extended the vocabulary for representing robust design decisions and specializations of engineering analysis models. Finally, the robustness of the language and DL implementation are illustrated through the extensibility,

consistency, and information organization. For example, as new concepts are added and/or existing concepts are modified changes are not propagated throughout the entire information base, but rather are localized for a particular concept. Holistically, the examples presented in this chapter demonstrate how the vocabulary, graphical information models, and DL representation are used to explicitly represent the information associated with engineering design decision and associated analysis models. Hence, we can say that empirical performance validity is achieved.

4.10 Chapter Synopsis

In this chapter a structured method for formulating design decisions is developed. The method is based on several design decision requirements and the general descriptors and keywords associated with compromise decision support problems (see Figure 4-33). The aims proposed at the beginning of this chapter are addressed:

- ✓ To introduce the information modeling requirements for modeling design decisions – A set of requirements are established specific to information modeling for engineering design decisions. These requirements are established through current literature and well-accepted methods and constructs for modeling engineering design decisions.
- ✓ To introduce a systematic method for formulating engineering design decisions - A systematic method is established for capturing the information associated with cDSPs. The method consists of seven phases for modeling the structure of decision information.

- ✓ To introduce the basic vocabulary (the syntax and semantics) for describing compromise decision support problems – A vocabulary for representing the semantics of cDSPs and analysis models is developed. The vocabulary is developed based on a critical evaluation of the cDSP construct.
- ✓ To illustrate how DL is used to specify several concept definitions of engineering design decisions – The vocabulary is used in conjunction DL for developing formal definitions of the cDSP, analysis models, and closely related concepts. The use of DL is demonstrated for specifying the information associated with several complex concepts. Additionally, DL reasoning algorithms are used to organize decision information.
- ✓ To critically assess the application of DL modeling in engineering design – DL is assessed for modeling engineering design information by logically applying and evaluating several example problems.

Information needed and generated and the relationships between the information is explicitly identified. The information is discussed in terms of arrays of mathematical representations. This information structure is abstracted and several concepts and properties are identified. Information models are then developed to explicitly represent the design decision information structure. Description Logic is then used to specify a formal language for describing decision information. Several decision problem formulations are presented using the language including the cDSP, MDO problems, and single and multi-objective specialization of these formulations.

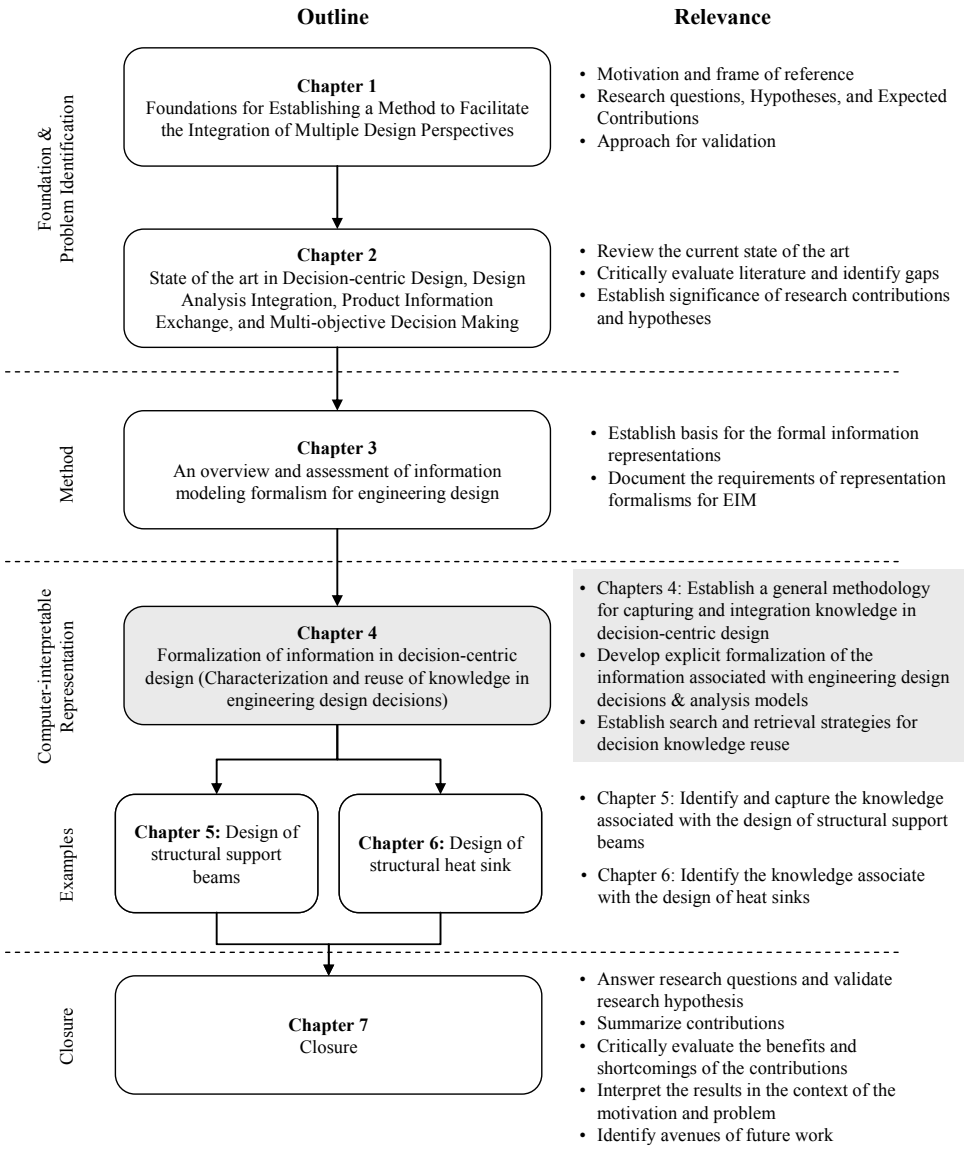


Figure 4-33: Outline of dissertation

The robust design decision formulation is presented to illustrate the extensibility of the language. For example, the robust formulation of the cDSP required additional properties that were not initially identified. The properties were added to the language without effecting the existing concept definitions. The representation of analysis model knowledge is then presented and represented using the formal language. Finally, the language is critically assessed and limitations are identified.

The foundational concepts for representing design decisions in a computational means are established in this chapter. The generic concepts specify the structure and information required for decision formulation and execution. In Chapter 5 and 6 the information models are used for representing specific design decisions for cantilever beams and structural heat sinks. The method for formulating design decisions is presented.

CHAPTER 5:

DESIGN OF STRUCTURAL BEAMS

Aims

- To demonstrate the use of the formal language for a simple, single-disciplinary design problem
- To capture the information structure associated with disciplinary analysis models

5.1 Problem Overview: Design of Structural Elements

The design of beams and columns are frequently encountered in civil and mechanical engineering design projects. A beam is a member subjected to loads applied in a transverse direction along the length of the beam. The applied load causes the member to bend. A cantilever beam configuration is illustrated in Figure 5-1.

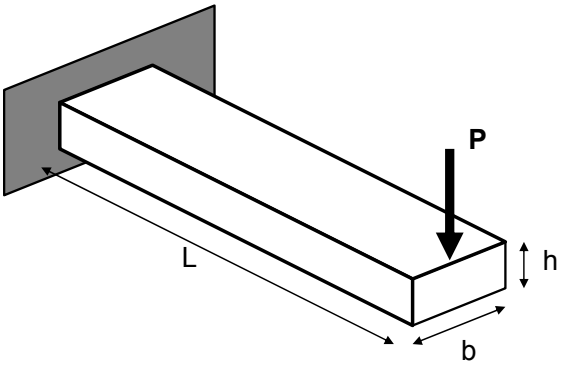


Figure 5-1: Cantilever beam configuration and loading

In this example, the design of cantilever beam involves the conflicting objectives of minimizing weight while simultaneously minimizing deflection at the free end. Additionally, the beam may be subjected to a number of constraints including buckling and stress. For this example, the information model and DL ontology are utilized to

explicitly capture the information associated with multi-objective decision-making with single disciplinary analysis models for analyzing the structural behavior of a cantilever beam. The cantilever beam design problem is a single disciplinary design problem (i.e., structural design) that involves several design objectives.

The cantilever beam examples are intended to demonstrate how the information model and DL-ontology can be used for (1) explicitly representing analysis model knowledge, (2) capturing limitations and assumptions of engineering analysis models (3) capturing the computational assumptions associated with computer simulations, (4) modeling multi-disciplinary engineering design decisions, and (5) reusing decision-related knowledge in the decision making process. A summary of the test plan and validation examples are presented in Table 5-1.

Table 5-1: Test plan and outline for cantilever beam

Step 1:	Identify the analysis support models for designing the cantilever beam. Derive the analysis relationships, assumptions, and computational limitations of the analysis models.
Step 2:	Capture the analysis model knowledge using the conceptual information model
Step 3:	Represent the analysis model knowledge computationally using the DL ontology
***Purpose*:**	Demonstrate the use of the conceptual information model and DL representation for explicitly capturing single disciplinary analysis models. Demonstrate that the formal language is extensible and robust, in that it can be used to model analysis models from multiple disciplines.

Step 4:	Formulate several multi-objective cantilever beam design problems that involve a variety of system goals, constraints and analysis models
Step 5:	Represent the decision-related knowledge using the conceptual information model
Step 6:	Implement the cantilever beam design decisions using the DL ontology.
***Purpose*:**	Demonstrate the use of the conceptual information model and DL representation for explicitly representing multi-objective design decisions. Demonstrate that the formal language is extensible and robust, in that it can be used to model decision information and facilitate integration.

5.2 Information Modeling of Cantilever Beam Analysis Models

The engineering analysis models are represented in accordance with the information model and DL language defined in Chapter 4.

5.2.1 Cantilever Beam Deflection Analysis Model

The analysis model relationships for computing the deflection of the cantilever beam is given as:

$$\delta = \frac{4PL^3}{Ebh^3} \qquad \textbf{5.1}$$

The angular rotation of the beam is given as:

$$\theta = \frac{6PL^2}{Ebh^3} \qquad \textbf{5.2}$$

where δ is the deflection at the free end of the beam, θ is the angular rotation of the beam, P is the load applied to the free end, L is the length of the beam, E is the modulus of elasticity of the beam material, h is the height of the beam, and b is the width of the

beam. The assumptions and limitations associated with the analysis relationships include both qualitative and when possible quantitative representations:

- Long beam assumption - $L \geq 10b$
- Slender beam assumption - $L \geq 10h$
- Constant, rectangular cross section
- Linear, isotropic material properties (The material properties must obey Hooke's law $\sigma = E\varepsilon$ and have the same values in all directions)
- No lateral torsional buckling occurs - $\sigma < \sigma_{Critical}$
- No soft material (The material properties must be sufficient high)
- No torsion in beam - $T = 0$
- Planes remain plane
- Small deflection assumption - $\theta \approx sin\,\theta$
- Elastic stress / deflection - $\sigma \leq \sigma_Y$
- Stress remains within elastic limits - $\sigma < \sigma_Y$

Two cantilever beam deflection analysis models are specified using the graphical information model and the DL-based representation. The underlying analysis model relationship is identical for each of the analysis models. However, the analysis models differ in the assumptions and limitations that are explicitly represented with the model. The cantilever beam analysis relationship for rotation is specified as:

```
Class(CantileverBeamEndLoadRotationRelationship partial
and(
EquationBasedConstraintRelationship
(∃ function_of_quantity.BeamWidth)
(∃ function_of_quantity.BeamLength)
(∃ function_of_quantity.ModulusofElasticity)
(∃ function_of_quantity.BeamHeight)
(∃ function_of_quantity.BeamRotation)
(∃ function_of_quantity.Point_Load)
(∀function_of_quantity.(BeamRotation ∪ Beam_Height ∪ Beam_Length ∪
  Beam_Width ∪ Modulus_of_Elasticity ∪ Point_Load))))
```

The deflection at the free end of the beam is specified as:

```
Class(CantileverBeamEndLoadDeflectionRelationship partial
and(
EquationBasedConstraintRelationship
(∃ function_of_quantity.BeamWidth)
(∃ function_of_quantity.BeamLength)
(∃ function_of_quantity.ModulusofElasticity)
(∃ function_of_quantity.BeamHeight)
(∃ function_of_quantity.BeamDeflection)
(∃ function_of_quantity.Point_Load)
(∀function_of_quantity.(BeamDeflection ∪ Beam_Height ∪ Beam_Length ∪
  Beam_Width ∪ Modulus_of_Elasticity ∪ Point_Load))))
```

The `EquationBasedConstraintRelationship` concept is used for modeling the deflection and the rotation of the cantilever beam. The cantilever beam deflection analysis model is represented using the afore-mentioned analysis relationships and is defined as:

```
Class(CantileverBeamEndLoadDeflectionAnalysisModel complete
and(
(∃ analysisrelationship.CantileverBeamEndLoadDeflectionRelationship)
(∃ analysisrelationship.CantileverBeamEndLoadRotationRelationship)
(∀ analysisrelationship.(CantileverBeamEndLoadDeflectionRelationship ∪
  CantileverBeamEndLoadRotationRelationship))
SubClassOf(CantileverBeamEndLoadDeflectionAnalysisModel
  EquationBasedAnalysisModel))
```

The second cantilever beam analysis model that predicts the vertical and rotational deflection at the free end of the beam uses the same analysis relationship, but differs in that several meta-level relationships are specified to define the validity space of the analysis relationships. The following meta-level relationships are specified for analysis models derived from general beam theory.

```
Class(long_beam_assumption partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamWidth)
  (∃ function_of_quantity.BeamLength)
  (∀function_of_quantity.(BeamLength ∪ BeamWidth))))
```

```
Class(slender_beam_assumption partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamHeight)
  (∃ function_of_quantity.BeamLength)
  (∀function_of_quantity.(BeamLength ∪ BeamHeight))))
```

```
Class(constant_cross_section_assumption partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamWidth)
  (∃ function_of_quantity.BeamHeight)
  (∀function_of_quantity.(BeamHeight ∪ BeamWidth))))
```

```
Class(constant_isotropic_material_properties partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamDensity)
  (∃ function_of_quantity.BeamYieldStrength)
  (∃ function_of_quantity. ModulusofElasticity)
  (∀function_of_quantity.(BeamDensity ∪ BeamYieldStrength ∪
  ModulusofElasticity))))
```

```
Class(no_lateral_torsional_buckling_occurs partial
and(
  EquationBasedConstraintRelationship
       (∃ function_of_quantity.LTBCriticalLoad)
  (∃ function_of_quantity. BeamNormalStress)
  (∀ function_of_quantity.(LTBCriticalLoad ∪ BeamNormalStress))))
```

```
Class(no_soft_material partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamYieldStrength)
  (∃ function_of_quantity. ModulusofElasticity)
  (∀function_of_quantity.(BeamYieldStrength ∪ ModulusofElasticity))))
```

```
Class(no_torsion_in_beam partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamTorsion)
  (∀function_of_quantity.BeamTorsion)))
```

```
Class(small_deflection partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamDeflection)
  (∀function_of_quantity.BeamDeflection)))
```

The limitations and assumptions are defined as meta-level `EquationBasedConstraintRelationship` concepts. Mathematically, the meta-level limitations and assumptions are identical to the analysis relationships. However, as the name implies the meta-level relationships govern when the analysis relationships can be used. The Euler cantilever beam analysis model captures the limitations explicitly and is represented as:

```
Class(EulerCantileverBeamEndLoadDeflectionAnalysisModel complete
and(
   (∃ analysisrelationship.CantileverBeamEndLoadDeflectionRelationship)
   (∃ analysisrelationship.CantileverBeamEndLoadRotationRelationship) ∩
   (∀ analysisrelationship.(CantileverBeamEndLoadDeflectionRelationship
   ∪ CantileverBeamEndLoadRotationRelationship))
   (∃ analysismetarelationship.long_beam_assumption))
   (∃ analysismetarelationship.slender_beam_assumption))
   (∃ analysismetarelationship.constant_cross_section_assumption)
   (∃ analysismetarelationship.constant_isotropic_material_properties)
   (∃ analysismetarelationship.no_lateral_torsional_buckling_occurs)
   (∃ analysismetarelationship.no_soft_material)
   (∃ analysismetarelationship.no_torsion_in_beam)
   (∃ analysismetarelationship.planes_remain_planes)
   (∃analysismetarelationship.small_deflection)
   (∃analysismetarelationship.elastic_stress)
   (∀analysismetarelationship.(constant_cross_section_assumption ∪
   constant_isotropic_material_properties ∪ small_deflection ∪
   long_beam_assumption ∪ no_lateral_torsional_buckling_occurs ∪
   no_soft_material ∪ no_torsion_in_beam ∪ planes_remain_planes ∪
   slender_beam_assumption))
SubClassOf(EulerCantileverBeamEndLoadDeflectionAnalysisModel
   EquationBasedAnalysisModel))
```

Graphical representations of the cantilever beam deflection analysis models are presented in Figure 5-2 and Figure 5-3. The graphical information illustrates the network of connections between analysis quantities, analysis relationships, and the meta-relationships. Additionally, the chain variables with other analysis models are identified.

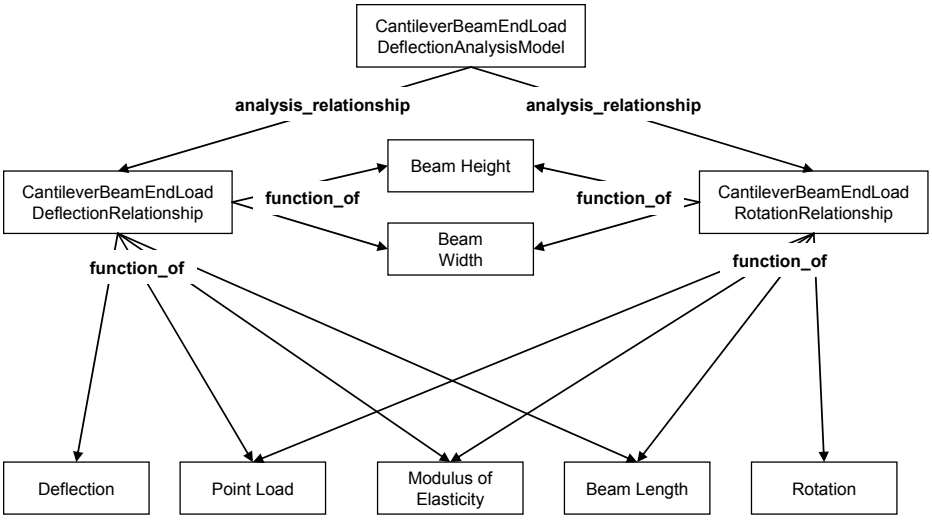


Figure 5-2: Graphical representation of cantilever beam deflection model

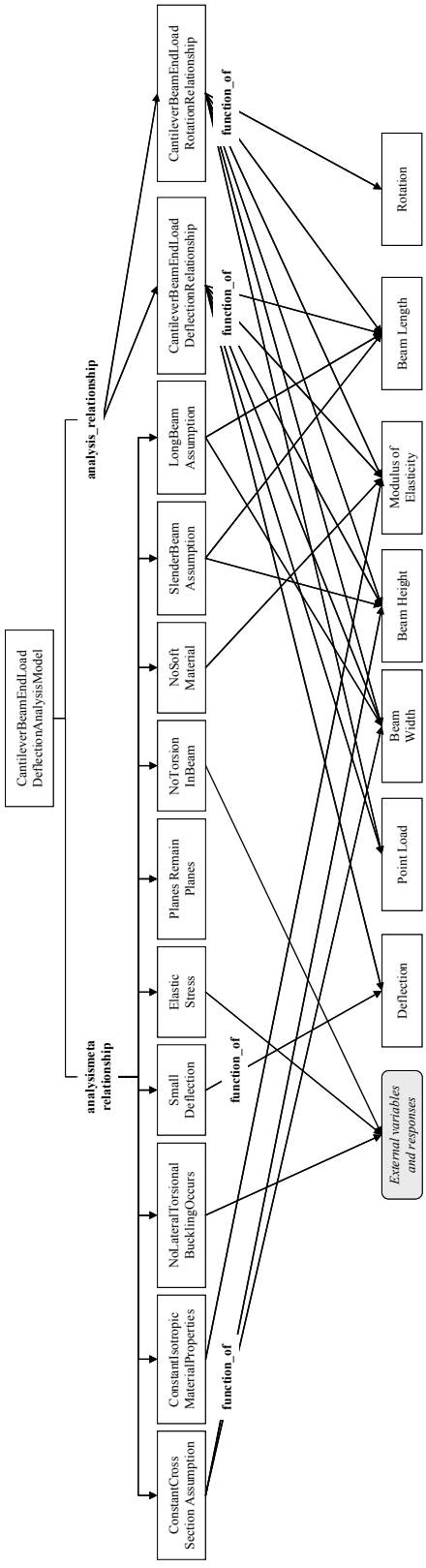


Figure 5-3: Graphical representation of Euler cantilever beam deflection model

As shown in Figures 5-2 and 5-3, the information associated with the cantilever beam deflection model and the Euler cantilever beam deflection model is different. The information required for using the analysis model is captured as a digital interface. For example, the aggregation of `Quantity` concepts associated with the deflection analysis models are the digital interface through which decision-related information is exchanged and analysis models are integrated into various design decisions. As shown in Figures 5-2 and 5-3, the digital interface is composed of the following specialized `Quantity` concepts: `Deflection`, `Rotation`, `BeamHeight`, `BeamWidth`, `BeamLength`, `PointLoad`, and `ModulusOfElasticity`. In addition, each graphical representation contains the same two analysis relationships for determining the vertical and rotational deflection in the beam, namely the `CantileverBeamEndLoadDeflectionRelationship` and `CantileverBeamEndLoadRotationRelationship`. However, in Figure 5-3 several additional meta-level relationships are specified that capture the limitations and constraints of the analysis model. The model presented in Figure 5-2 is not "wrong" however the representation is *less rich* than the model presented in Figure 5-3. For example, explicitly capturing the meta-level relationships provides a means for determining the validity of the design solutions. However, a result of increasing the richness of the model is imposing additional information requirements. For example, the `ElasticStress` assumption captured in Figure 5-3 requires that deflection model can be used when the stresses in the beam are within the elastic limit. The maximum stress in the beam must be computed via an external analysis model and the results propagated to the deflection model (see Section 5.2.2 for the derivation of the stress model).

Additionally, the deflection model is valid when lateral torsional buckling (i.e., `no_lateral_torsional_buckling_occurs`) in the beam does not occur. The critical load that causes lateral torsional buckling in the beam must first be computed in an external analysis model, which is then used to determine if lateral torsional buckling occurs. Thus, several models are required and must be chained together to ensure the assumptions associated with the deflection analysis model are satisfied. In the next section, the analysis model for computing the stress in the beam is derived. A discussion is presented in Section 5.5 based on the implementation and representation of several analysis models.

5.2.2 Cantilever Beam Stress Analysis Model

The relationship for computing the stress in the cantilever beam is given as:

$$\sigma = \frac{6PL}{bh^2} \qquad \textbf{5.3}$$

where σ is the normal stress along the length of the beam, P is the load applied to the free end, L is the length of the beam, h is the height of the beam and b is the width of the beam. The same assumptions and limitations associated with the deflection analysis model are specified for the stress analysis models. The cantilever beam analysis relationship for stress is specified as:

```
Class(CantileverBeamEndLoadStressRelationship partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamWidth)
  (∃ function_of_quantity.BeamLength)
  (∃ function_of_quantity.BeamHeight)
  (∃ function_of_quantity.BeamNormalStress)
  (∃ function_of_quantity.Point_Load)
  (∀function_of_quantity.(BeamNormalStress ∪ Beam_Height ∪ Beam_Length
  ∪ Beam Width ∪ Point Load))))
```

The cantilever beam analysis models use the previously defined relationship and meta-level relationships. The general cantilever beam stress analysis model is defined as:

```
Class(CantileverBeamEndLoadStressAnalysisModel complete
and(
  (∃ analysisrelationship.CantileverBeamEndLoadStressRelationship)
  (∀ analysisrelationship.(CantileverBeamEndLoadStressRelationship)
SubClassOf(CantileverBeamEndLoadDeflectionAnalysisModel
  EquationBasedAnalysisModel))
```

The meta-level relationships derived for the deflection analysis model are reused for specifying the validity space for the stress analysis model. The analysis model for computing stress is the cantilever beam is defined as:

```
Class(EulerCantileverBeamEndLoadStressAnalysisModel complete
and(
   (∃ analysisrelationship.CantileverBeamEndLoadStressRelationship)
   (∀ analysisrelationship.(CantileverBeamEndLoadStressRelationship)
   (∃ analysismetarelationship.long_beam_assumption)
   (∃ analysismetarelationship.slender_beam_assumption)
   (∃ analysismetarelationship.constant_cross_section_assumption)
   (∃ analysismetarelationship.constant_isotropic_material_properties)
   (∃ analysismetarelationship.no_lateral_torsional_buckling_occurs)
   (∃ analysismetarelationship.no_soft_material)
   (∃ analysismetarelationship.no_torsion_in_beam)
   (∃ analysismetarelationship.planes_remain_planes)
   (∃analysismetarelationship.small_deflection)
   (∃analysismetarelationship.elastic_stress)
   (∀analysismetarelationship.(constant_cross_section_assumption ∪
   constant_isotropic_material_properties ∪ small_deflection ∪
   long_beam_assumption ∪ no_lateral_torsional_buckling_occurs ∪
   no_soft_material ∪ no_torsion_in_beam ∪ planes_remain_planes ∪
   slender_beam_assumption ∪ elastic_stress))
SubClassOf(EulerCantileverBeamEndLoadStressAnalysisModel
   EquationBasedAnalysisModel))
```

Graphical representations of the cantilever beam stress analysis models are presented in Figure 5-4 and Figure 5-5.

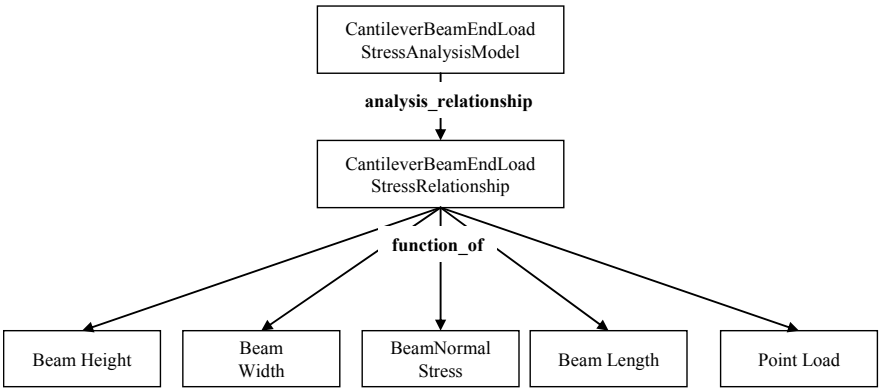


Figure 5-4: Graphical representation of cantilever beam stress model

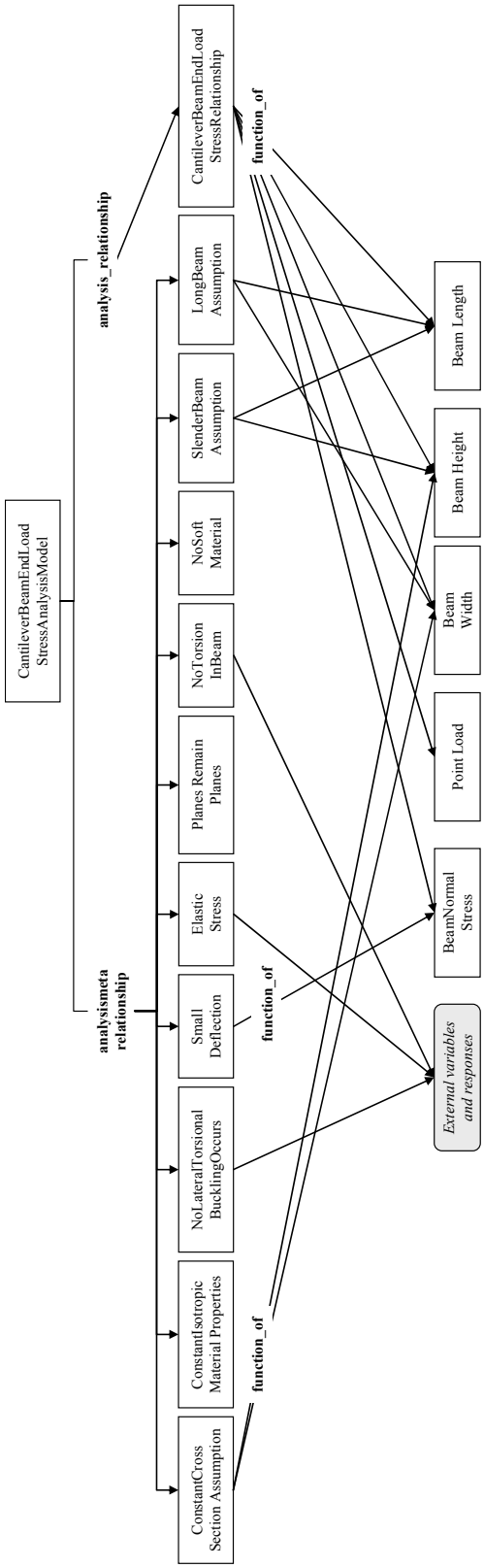


Figure 5-5: Graphical representation of Euler cantilever beam stress model

The information associated with the cantilever beam stress models is illustrated graphically in Figures 5-4 and 5-5. A similar discussion is presented for the stress analysis models as for the deflection analysis models. The digital interface for the stress models is defined by the following Quantity concepts: `BeamNormalStress`, `BeamHeight`, `BeamWidth`, `BeamLength`, and `PointLoad`. The analysis model illustrated in Figure 5-5 is richer than that presented in Figure 5-4 through the inclusion of several limitations and assumptions. Additionally, the meta-constraints that define the validity space of the stress analysis model represented in Figure 5-5 are identical to the assumptions and limitation of the deflection model. This is expected because the models are derived using beam theory and the same governing equation. Thus, several of the `ConstraintRelationship` concepts defined in Section 5.2.1 can be reused for specifying the validity space of the stress analysis model. For example, the `constant_cross_section_assumption`, `constant_isotropic_material_properties`, `no_lateral_torsional_buckling_occurs`, `small_deflection`, `long_beam_assumption`, `no_soft_material`, `no_torsion_in_beam`, `planes_remain_planes`, `slender_beam_assumption`, and `elastic_stress` assumptions are reused across the deflection and stress analysis models. The cantilever beam stress model represented in Figure 5-5 requires information to be instantiated and exchanged through several analysis model chains. For example, the same chain between the stress analysis model and the lateral torsional buckling analysis model must be established to ensure the `no_lateral_torsional_buckling_occurs` assumption is not violated.

The graphical representations of the beam stress model presented in Figures 5-4 and 5-5 enable engineering decision makers to quickly gain an understanding of the limitations of the model, the information required to use the models, and the dependencies between the models.

5.2.3 Beam Weight Analysis Model

The relationship for computing the weight in the cantilever beam is given as:

$$w = bhL\rho g \qquad \textbf{5.4}$$

where w is the weight, L is the length of the beam, h is the height of the beam, b is the width of the beam, ρ is the density of the beam material, and g is gravity. The assumptions and limitations associated with the analysis relationship include the following:

- Constant, rectangular cross section
- Homogeneous material properties

The cantilever beam analysis relationship for stress is specified as:

```
Class(BeamWeightRelationship partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamWidth)
  (∃ function_of_quantity.BeamLength)
  (∃ function_of_quantity.BeamHeight)
  (∃ function_of_quantity.BeamDensity)
  (∃ function_of_quantity.BeamWeight)
  (∃ function_of_quantity.Gravity)
  (∀function_of_quantity.(BeamWeight ∪ Beam_Height ∪ Beam_Length ∪
  Beam_Width ∪ BeamDensity ∪ Gravity))))
```

The beam weight analysis model uses the previously defined relationship and meta-level relationships and is given as:

```
Class(BeamWeightAnalysisModel complete
and(
   (∃ analysisrelationship.BeamWeightRelationship)
   (∀ analysisrelationship.BeamWeightRelationship)
SubClassOf(BeamWeightAnalysisModel EquationBasedAnalysisModel)
```

The constrained cantilever beam weight analysis model captures the limitations explicitly and is represented as:

```
Class(ConstrainedBeamWeightAnalysisModel complete
and(
   (∃ analysisrelationship.BeamWeightRelationship)
   (∀ analysisrelationship.BeamWeightRelationship)
   (∃ analysismetarelationship.constant_cross_section_assumption)
   (∃ analysismetarelationship.constant_material_properties)
   (∀analysismetarelationship.(constant_cross_section_assumption ∪
   constant_material_properties))
SubClassOf(BeamWeightAnalysisModel EquationBasedAnalysisModel))
```

Graphical representations of the beam weight analysis model are presented in Figure 5-6.

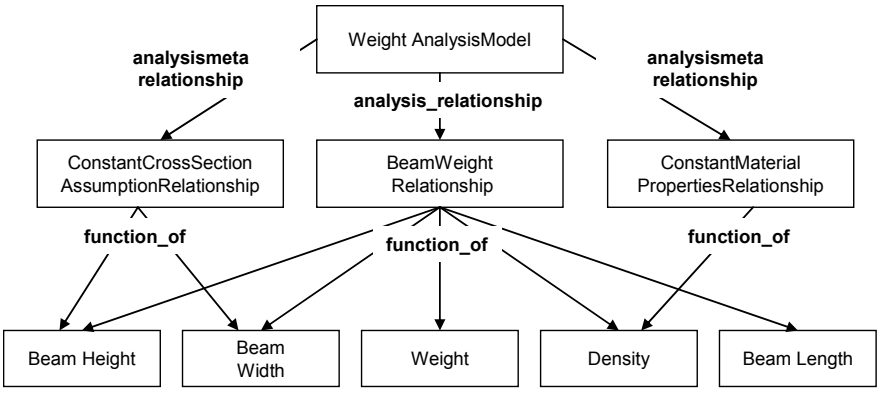


Figure 5-6: Graphical representation of weight analysis model

As illustrated in Figure 5-6, the `WeightAnalysisModel` has two meta-level relationships that limit the applicability of the analysis model and a single analysisrelationship for computing the weight of the beam. The meta-level analysis models specify that the cross sectional area of the beam must be constant and the material properties of the beam must be uniform. The analysis meta-relationships can be reused from the concept definitions established in Section 5.2.1. The graphical representation illustrates the model is not *chained* to other analysis models, but rather several internal meta-relationships must be checked. For example, the analysis model is applicable for constant, rectangular cross section and constant density. In other words, the analysis model does not produce valid results if the cross section and the material density are not constant.

5.2.4 Cantilever Beam Lateral Torsional Buckling

When beams are subjected to a transverse load that causes flexure about the primary axis, they are prone to lose stability and buckle due to a *twisting* about the weaker axis. Detailed derivations of lateral torsional buckling can be found in [12, 142]. The relationships for computing the weight in the cantilever beam are given as [142].

$$J = \frac{1}{12} bh \cdot \left(b^2 + h^2 \right) \qquad \textbf{5.5}$$

$$L_{effective} = \frac{0.783L}{1 - \frac{2h}{L}} \qquad \textbf{5.6}$$

$$\alpha^2 = \sqrt{2\pi} \cdot \frac{EI}{\sqrt[4]{GJ}} \cdot \frac{\sqrt{L_{effective} h}}{b} \qquad \textbf{5.7}$$

$$F_{Critical} = \frac{\pi^2 E}{\alpha^2} \qquad \textbf{5.8}$$

where b is the beam width, h is the beam height, L is beam length, G is the shear modulus, and E is the modulus of elasticity. The cantilever beam lateral torsional buckling relationship is specified as:

```
Class(CantileverBeamLTBRelationship partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.BeamWidth)
  (∃ function_of_quantity.BeamLength)
  (∃ function_of_quantity.BeamHeight)
  (∃ function_of_quantity.LTBCriticalLoad)
  (∃ function_of_quantity.ModulusofElasticity)
  (∃ function_of_quantity.ShearModulus)
  (∀ function_of_quantity.(BeamWidth ∪ BeamHeight ∪ BeamLength ∪
  LTBCriticalLoad ∪ ModulusofElasticity ∪ ShearModulus))))
```

The lateral torsional buckling analysis model is defined as:

```
Class(CantileverBeamLTBModel complete
and(
   (∃ analysisrelationship.CantileverBeamLTBRelationship)
   (∀ analysisrelationship.CantileverBeamLTBRelationship)
SubClassOf(CantileverBeamLTBModel EquationBasedAnalysisModel)))
```

Several of the meta-level relationships are reused for specifying the validity space of the lateral torsional buckling analysis model with the exception of the *"No lateral torsional buckling occurs"* limitation. The lateral torsional buckling analysis model considering meta-level constraints is defined:

```
Class(EulerCantileverBeamLTBModel complete
and(
   (∃ analysisrelationship.CantileverBeamLTBRelationship)
   (∀ analysisrelationship.CantileverBeamLTBRelationship)
   (∃ analysismetarelationship.long_beam_assumption)
   (∃ analysismetarelationship.slender_beam_assumption)
   (∃ analysismetarelationship.constant_cross_section_assumption)
   (∃ analysismetarelationship.constant_isotropic_material_properties)
   (∃ analysismetarelationship.no_soft_material)
   (∃ analysismetarelationship.no_torsion_in_beam)
   (∃ analysismetarelationship.planes_remain_planes)
   (∃ analysismetarelationship.small_deflection)
   (∃ analysismetarelationship.elastic_stress)
   (∀ analysismetarelationship.(constant_cross_section_assumption ∪
   constant_isotropic_material_properties ∪ small_deflection ∪
   long_beam_assumption ∪ no_soft_material ∪ no_torsion_in_beam ∪
   planes_remain_planes ∪ slender_beam_assumption ∪ elastic_stress))
SubClassOf(EulerCantileverBeamLTBModel EquationBasedAnalysisModel)))
```

Graphical representations of the lateral torsional analysis models are presented in Figure 5-7 and Figure 5-8.

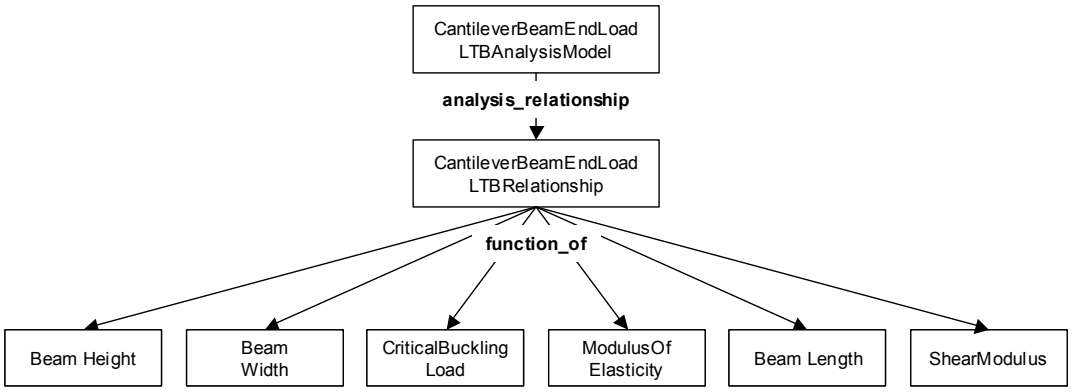


Figure 5-7: Graphical representation of lateral torsional buckling analysis model

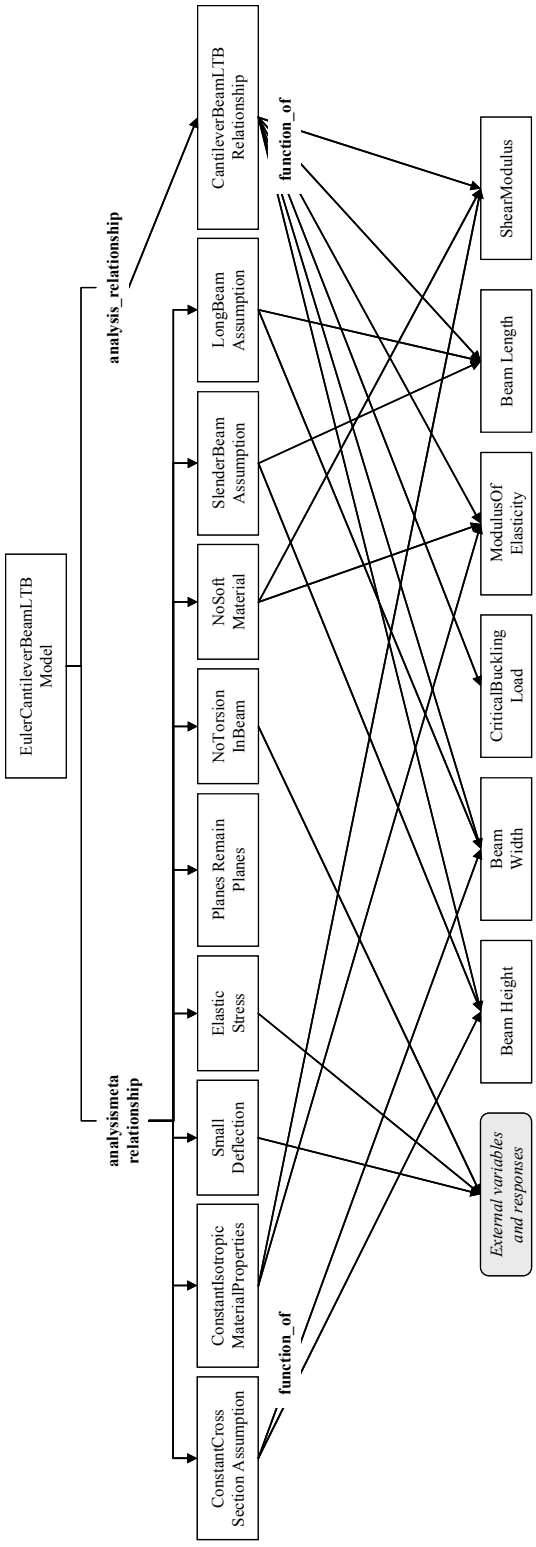


Figure 5-8: Graphical representation of cantilever beam lateral torsional buckling analysis model with meta-level constraints

Similar to the previously discussed analysis models, two models are represented for determining the critical load lateral torsional buckling that occurs. In the first model, only the analysis relationship is captured, while in the second model in Figure 5-8 the analysis relationship and several meta-level relationships are represented. The specification of meta-level relationships imposes additional requirements on the development of the analysis model, but results in decreased misuse. The analysis model for computing the lateral torsional buckling critical stress is constrained by nine limiting assumptions. These meta-level constraint relationships are the same as those used in the deflection and stress analysis models and thus can be reused. For example, as illustrated in Figure 5-8 and in the DL representation, the lateral torsional buckling analysis models is restricted by several assumptions including: `constant_cross_section_assumption`, `constant_isotropic_material_properties`, `small_deflection`, `long_beam_assumption`, `no_soft_material`, `no_torsion_in_beam`, `planes_remain_planes`, `slender_beam_assumption`, and `elastic_stress`. The digital interface for the lateral torsional buckling model is comprise of the following `Quantity` concepts: `BeamWidth`, `BeamHeight`, `BeamLength`, `LTBCriticalLoad`, `ModulusofElasticity`, and `ShearModulus`. However, the analysis model represented in Figure 5-8 requires additional information to be instantiated from other analysis models and information sources. For example, the yield stress of the material must be known and the actual stress in the beam must be computed to verify the `elastic_stress` assumption is satisfied. Similarly, the deflection in the beam must be computed to check the `small_deflection` assumption. The graphical representation enables the

information associated with the lateral torsional buckling analysis model to be visualized and the required information and analysis models to be identified. The graphical representations provide the conceptual schematic on which the DL representation is developed.

In Sections 5.2.1 through 5.2.4 the information associated with several engineering analysis models for determining the behavior of a cantilever beam are presented. These analysis models are used to support engineering design decisions. In the following sections, the integration of the analysis models is illustrated in two cDSPs, followed by a discussion about the value for developing DL representation of analysis and decision information.

5.3 Information Modeling of Cantilever Beam Design Problem

Two different formulations of the cantilever beam design problem are developed. The formulation uses analysis models that do not explicitly capture the limitations and assumptions, while the second cantilever beam design problem takes into account analysis constraint.

5.3.1 Cantilever Beam Design Problem Example 1

The cantilever beam design problem is summarized in Table 5-2.

Table 5-2: Cantilever beam design problem description (no analysis constraints)

- The system variables include: beam width, beam height
- The design parameters include: beam length, material properties, load on beam,
- The target deflection and target weight of the beam are known
- The design objectives are (1) minimize deflection of the free and (2) minimize the weight of the beam
- No analysis constraints are taken into account.

The beam design problem is modeled as a cDSP (see Figure 5-9).

GIVEN

Design parameters

- Applied distributed load, $P = 150\,N$
- Beam length, $L = 0.5\,m$
- Gravity, $g = 9.81 \mathrm{m/sec^2}$
- Modulus of elasticity, $E = 205\,GPa$
- Density, $\rho = 7872\,kg/m^3$

Disciplinary Analysis Models

- Structural analysis models
  1. Vertical displacement caused by transverse loading, $\delta = f(P, L, E, b, h)$
  2. Weight of beam, $W = f(\rho, L, b, h, g)$

FIND

System variables

- Beam width
- Beam height

Deviation variables

- Deviation from target weight
- Deviation from target deflection

SATISFY

System bounds

- Bounds on beam width, $0.01m \le b \le 0.1m$
- Bounds on beam height, $0.01m \le h \le 0.1m$

System design requirements & constraints

- Beam width is positive value, $b > 0$
- Beam height is positive value, $h > 0$

Analysis models constraints & limitations

- *No constraints imposed from the analysis models*

System Goals

- Minimize deviation of beam deflection from target deflection, $\delta = f(b, h, L, E, P)$
- Minimize deviation of beam weight from target weight, $W_{Critical} = f(b, h, \rho, g, L)$

Decision constraints

- $d_i^+, d_i^- \ge 0$ & $d_i^+ \cdot d_i^- = 0$ for all design objectives

MINIMIZE

Deviation function Archimedean formulation

$Z = f\left(w_{weight} \cdot d_{weight}^- + w_{deflection} \cdot d_{deflection}^-\right)$

Figure 5-9: Cantilever design problem 1 – analysis constraints not considered in decision formulation

A graphical illustration of the cantilever beam design problem 1 is presented in Figure 5-10.

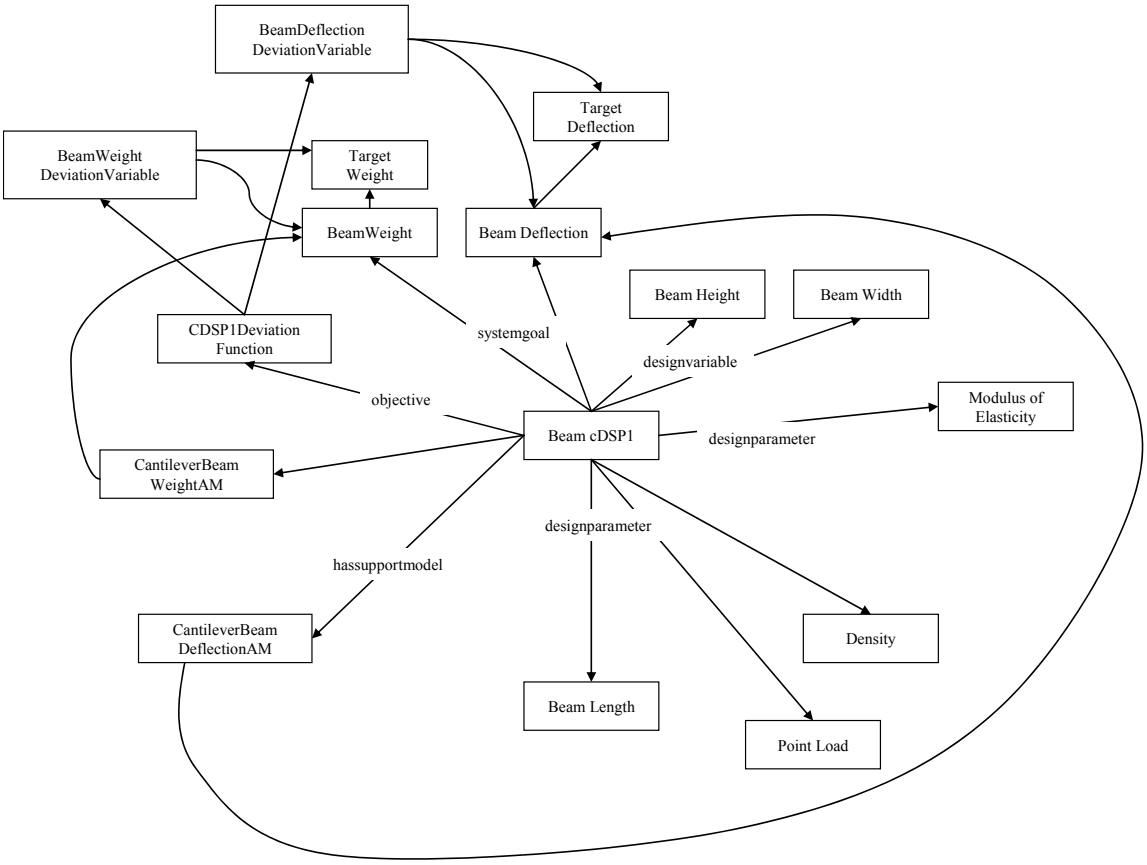


Figure 5-10: Graphical representation of cantilever beam problem 1

Figure 5-10 is a simplified view of the total set of information associated with the cantilever beam design problem 1. For example, the relationships between the design variables and design parameters for all of the analysis models and the design decision are not captured in Figure 5-10. As illustrated in Figure 5-10, the analysis models defined in Sections 5.2.1 through 5.2.4 are integrated with the decision-related information by explicitly establishing linkages between the cDSP concepts and the appropriate analysis models. The detailed mapping between the analysis model quantities and the decision quantities are not captured in Figure 5-10. A subset of the relationships is illustrated for computing the deflection of the beam. The deflection of the beam is computed in the CantileverBeamDeflectionAM, which is a function of BeamLength, BeamWidth, and

BeamHeight, and PointLoad. These quantities are instantiated in the cDSP model and the relationships are represented in Figure 5-11.

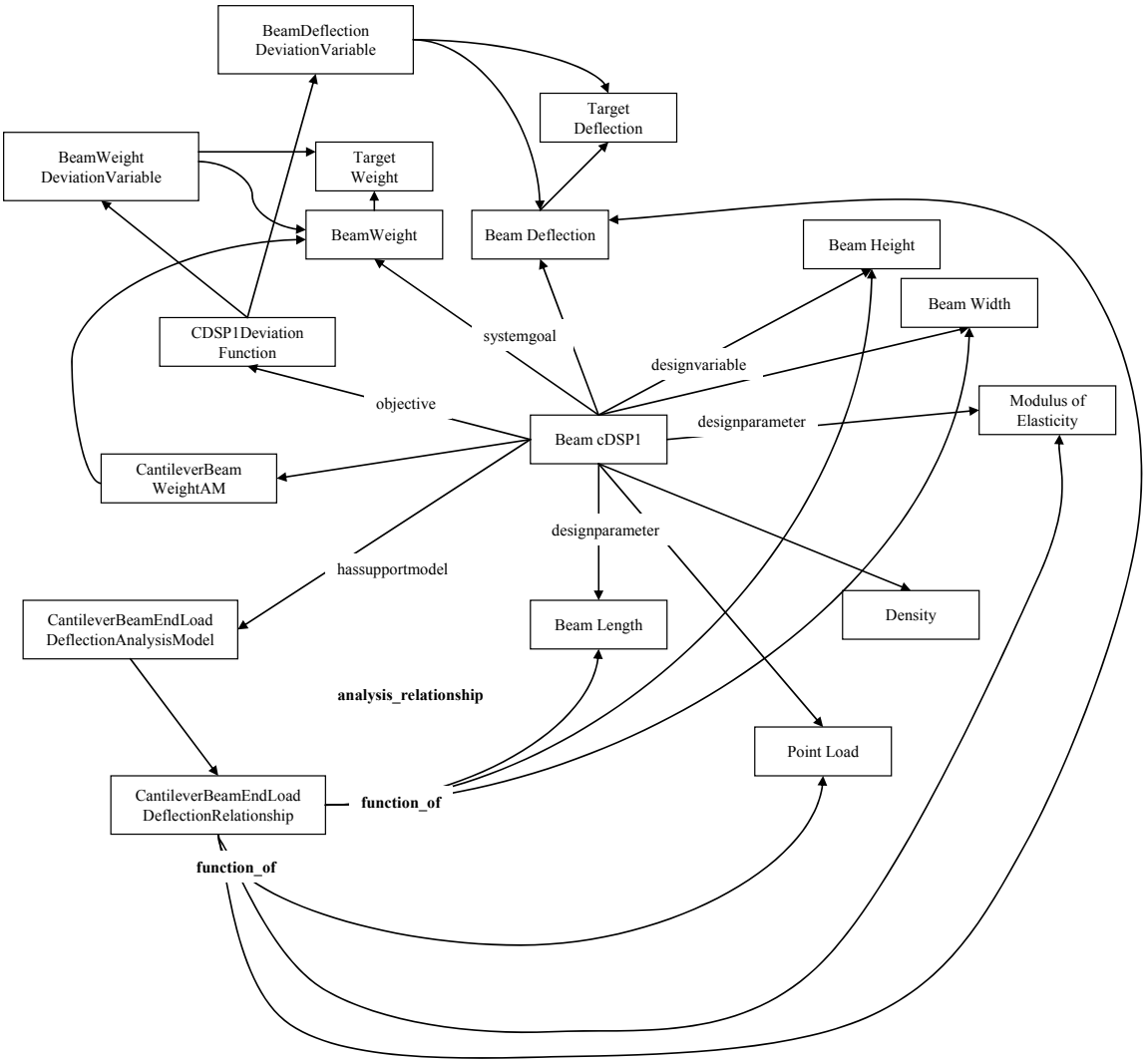


Figure 5-11: Integration and mapping of Quantity concepts between analysis models and the cDSP for cantilever beam design problem 1

As shown in Figure 5-11, the quantities associated with the analysis model and the corresponding quantities with the cDSP are associated. A similar process is repeated for each of the analysis models. The aggregation of the "linkages" between the analysis model concept and Quantity concepts in the cDSP represent the interface between the

models. The cDSP representation captures the dependencies between analysis models, the drivers for determining the deviation function of the cDSP, and the information required to execute the cDSP. Based on the graphical representation, a DL-based implementation is developed.

```
Class (CantileverBeamCDSP1
and (
 (∃ designvariable.(BeamWidth∩(=upperbound 1)∩(=lowerbound 1)))
 (∃ designvariable.(BeamHeight∩(=upperbound 1)∩(=lowerbound 1)))
 (= designvariable 2)

 (∃ designparameter.BeamLength)
 (∃ designparameter.PointLoad)
 (∃ designparameter.ModulusofElasticity)
 (∃ designparameter.Density)
 (∃ designparameter.Gravity)

 (∃ hassupportmodel.CantileverBeamWeightAM)
 (∃ hassupportmodel.CantileverBeamDeflectionAM)
 (≥ hassupportmodel 1)

 (∃ systemgoal.(BeamWeight ∩
   (= targetsystembehavior 1) ∩
   (∃ targetsystembehavior.TargetWeight) ∩
   (∀ targetsystembehavior. TargetWeight)))

 (∃ systemgoal.(BeamDeflection ∩
   (= targetsystembehavior 1) ∩
   (∃ targetsystembehavior.TargetDeflection) ∩
   (∀ targetsystembehavior.TargetDeflection)))
 (= systemgoal 2)
 (∃ objective.(CDSP1DeviationFunction ∩
   (∃ function_of.(BeamWeightDeviationVariable ∩
    ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
       ((∃ function_of.BeamWeight) ∩
       (∃ function_of.TargetWeight) ∩
       (∀ function_of.(BeamWeight∪TargetWeight))) ∩
   (∃ function_of.(BeamDeflectionDeviationVariable ∩
    ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
       ((∃ function_of. BeamDeflection) ∩
       (∃ function_of. TargetDeflection) ∩
       (∀ function_of.( BeamDeflection ∪ TargetDeflection))))
```

```
(∀ objective.(CDSP1DeviationFunction ∩
   (∃ function_of.(BeamWeightDeviationVariable ∩
    ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩
      ((∃ function_of.BeamWeight) ∩
      (∃ function_of.TargetWeight) ∩
      (∀ function_of.(BeamWeight∪TargetWeight))) ∩
   (∃ function_of.(BeamDeflectionDeviationVariable ∩
    ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩
      ((∃ function_of. BeamDeflection) ∩
      (∃ function_of. TargetDeflection) ∩
      (∀ function_of.(BeamDeflection ∪ TargetDeflection))))
  (= objective 1)
```

In the next section, the similar cantilever beam design problem is presented. However, different engineering analysis models are used to support the design decision. The information representations are presented followed by a discussion and detailed comparison. Finally, the value in developing computational representation based on DL is presented.

5.3.2 Cantilever Beam Design Problem Example 2

The second cantilever beam design problem is summarized in Table 5-3.

Table 5-3: Cantilever beam design problem description

- The system variables include: beam width, beam height
- The design parameters include: beam length, material properties, load on beam,
- The target deflection and target weight of the beam are known
- The design objectives are (1) minimize deflection of the free and (2) minimize the weight of the beam
- Analysis constraints are propagated from analysis models

The beam design problem is modeled as a cDSP (see Figure 5-12).

GIVEN

Design parameters

- Applied distributed load, $P = 150\,N$
- Beam length, $L = 0.5\,m$
- Gravity, $g = 9.81\,\mathrm{m/sec^2}$
- Modulus of elasticity, $E = 205\,GPa$
- Shear modulus, $G = 80\,GPa$
- Density, $\rho = 7872\,kg/m^3$
- Yield Strength, $S_Y = 340\,MPa$

Disciplinary Analysis Models

- Structural analysis models
  1. Euler Vertical displacement caused by transverse loading, $\delta = f(P, L, E, b, h)$
  2. Euler Angular displacement caused by transverse loading $\theta = f(P, L, E, b, h)$
  3. Euler Tensile stress along length of beam due to bending(transverse loading), $\sigma = f(P, L, b, h)$
  4. Weight of beam, $W = f(\rho, L, b, h, g)$
  5. Lateral torsional buckling critical stress, $\sigma_{Critical} = f(b, h, L, E, G)$

FIND

System variables

- Beam width
- Beam height

Deviation variables

- Deviation from target weight
- Deviation from target deflection

SATISFY

System bounds

- Bounds on beam width, $0.01m \le b \le 0.1m$
- Bounds on beam height, $0.01m \le h \le 0.1m$

System design requirements & constraints

- Beam width is positive value, $b > 0$
- Beam height is positive value, $h > 0$

Analysis models constraints & limitations

- *Imposed by analysis models selected*

System Goals

- Minimize deviation of beam deflection from target deflection, $\delta = f(b, h, L, E, P)$
- Minimize deviation of beam weight from target weight, $W_{Critical} = f(b, h, \rho, g, L)$

Decision constraints

- $d_i^+, d_i^- \ge 0$ & $d_i^+ \cdot d_i^- = 0$ for all design objectives

MINIMIZE

Deviation function Archimedean formulation

$$Z = f\left(w_{weight} \cdot d_{weight}^- + w_{deflection} \cdot d_{deflection}^-\right)$$

Figure 5-12: Cantilever design problem 2 – analysis constraints considered in decision formulation

The second cantilever beam design problem is represented graphically in Figure 5-13.

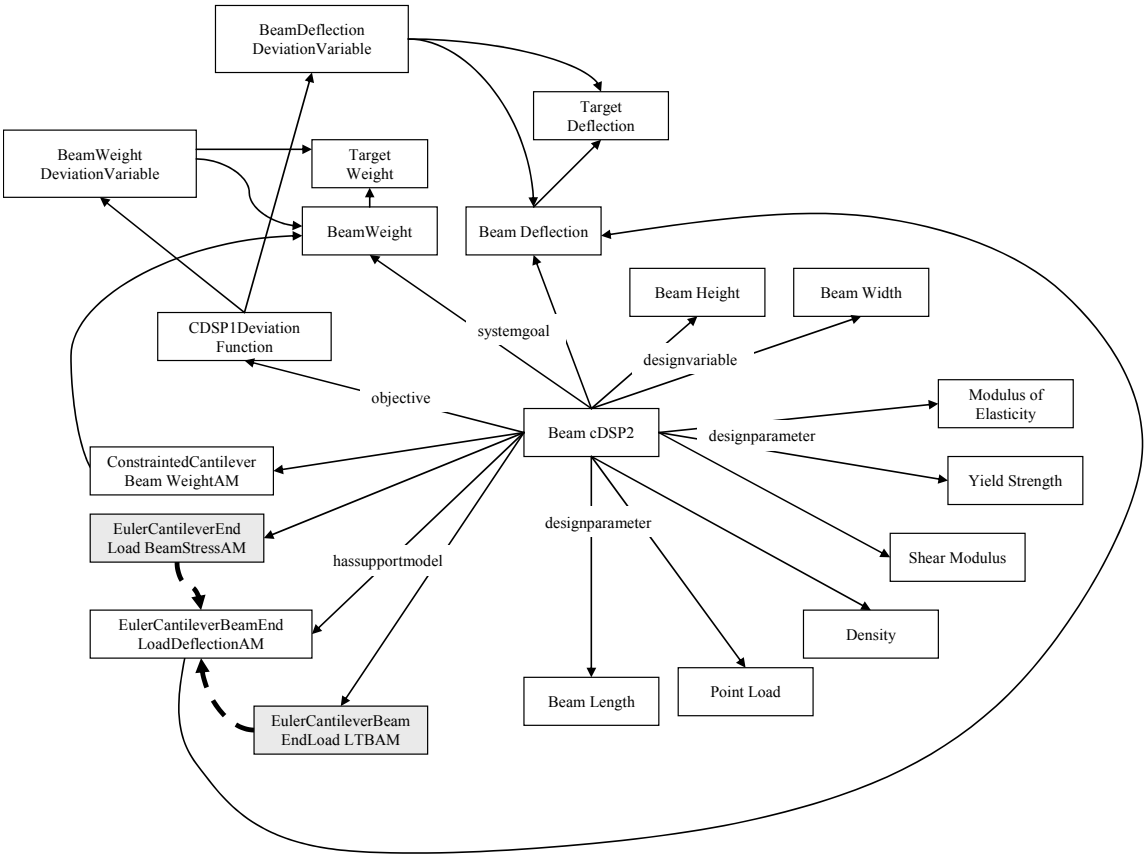


Figure 5-13: Graphical representation of cantilever beam problem 2

The graphical representation of the cantilever beam design problem 2 illustrates the chain relationships between the analysis models as a result of the meta-level constraint relationships. For example, the meta-level constraints associated with the `EulerCantileverBeamDeflectionAM` (Section 5.2.1) require that the maximum stress in the beam be computed in an external model (i.e., the `EulerCantileverBeamStressAM`) and the critical load that results in lateral torsional buckling (i.e., `EulerCantileverBeamLTBAM`). Thus, chain relationships must be established between the analysis models and additional information must be specified in the design decision. A detailed representation is presented in Figure 5-14.

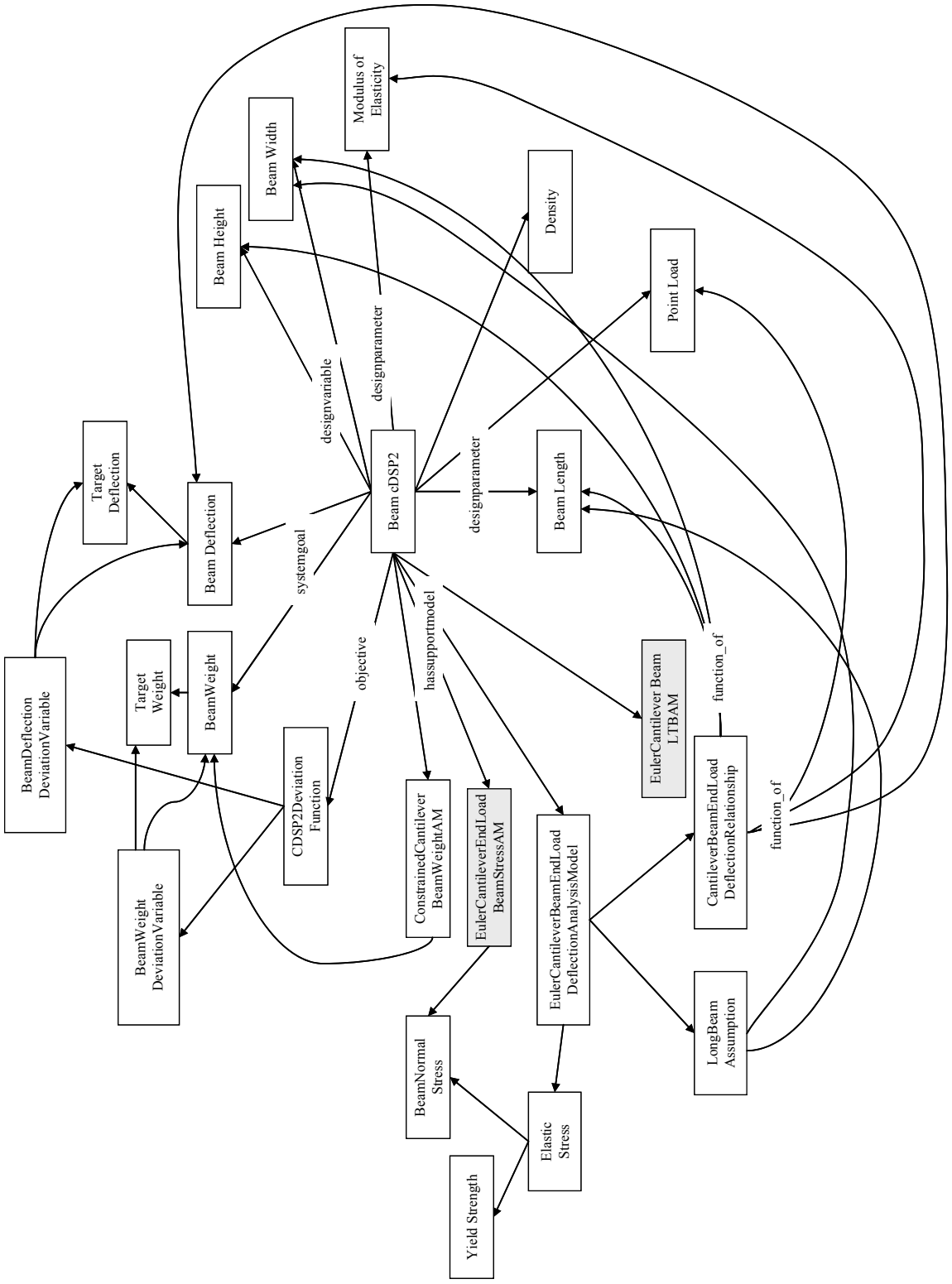


Figure 5-14: Integration and mapping of Quantity concepts between analysis models and the cDSP for cantilever beam design problem 2

As illustrated in Figure 5-14, the `EulerCantileverBeamEndLoadDeflectionAnalysisModel` is a function of `BeamLength`, `BeamWidth`, and `BeamHeight`, and `PointLoad`. In addition, the `elastic_stress` meta-level constraint that limits the analysis model is a function of `YieldStrength`, and `BeamNormalStress`. The yield strength of the material may be taken from a material database and thus can be instantiated in the design decision. However, the normal stress in the beam must be computed in an external analysis model. Thus, linkages between the deflection analysis model and the stress analysis models are defined based on common `Quantity` concepts. It is shown in Figure 5-14, that the analysis model are integrated into the design decision and information is shared through an established set of domain-specific vocabulary. The graphical representation of the cantilever beam design problem 2 capture the integration and exchange of information across design decisions and analysis models and provides a means for determining the dependency of analysis models based on meta-level relationships and analysis relationships to capture the complex coupling of system behaviors. The graphical representations of cantilever beam problem 1 (Figures 5-11 and 5-12) and cantilever beam design problem 2 (Figures 5-13 and 5-14) are very similar. In actuality, the entire set of information captured in problem 1 is represented in problem 2. In other words, design problem 2 is much richer than design problem 1. Thus, design problem 2 is a subclass of design problem 1. In a similar manner to design problem 1, a DL representation is derived based on the graphical information model. The cantilever beam design problem 2 is represented using the formal language as:

```
Class (CantileverBeamCDSP2
and (
  (∃ designvariable.(BeamWidth∩(=upperbound 1)∩(=lowerbound 1)))
  (∃ designvariable.(BeamHeight∩(=upperbound 1)∩(=lowerbound 1)))
  (= designvariable 2)

  (∃ designparameter.BeamLength)
  (∃ designparameter.PointLoad)
  (∃ designparameter.ModulusofElasticity)
  (∃ designparameter.Density)
  (∃ designparameter.ShearModulus)
  (∃ designparameter.YieldStrength)
  (∃ designparameter.gravity)

  (∃ hassupportmodel.CantileverBeamWeightAM)
  (∃ hassupportmodel.EulerCantileverBeamStressAM)
  (∃ hassupportmodel.EulerCantileverBeamDeflectionAM)
  (∃ hassupportmodel.EulerCantileverBeamLTBAM)
  (≥ hassupportmodel 1)

  (∃ systemgoal.(BeamWeight ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.TargetWeight) ∩
    (∀ targetsystembehavior.TargetWeight)))

  (∃ systemgoal.(BeamDeflection ∩
    (= targetsystembehavior 1) ∩
    (∃ targetsystembehavior.TargetDeflection) ∩
    (∀ targetsystembehavior.TargetDeflection)))
  (= systemgoal 2)

  (∃ objective.(CDSP1DeviationFunction ∩
    (∃ function_of.(BeamWeightDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩
         ((∃ function_of.BeamWeight) ∩
         (∃ function_of.TargetWeight) ∩
         (∀ function_of.(BeamWeight∪TargetWeight))) ∩
    (∃ function_of.(BeamDeflectionDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩
         ((∃ function_of. BeamDeflection) ∩
         (∃ function_of. TargetDeflection) ∩
         (∀ function_of.( BeamDeflection ∪ TargetDeflection))))
```

```
(∀ objective.(CDSP1DeviationFunction ∩
   (∃ function_of.(BeamWeightDeviationVariable ∩
    ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩
      ((∃ function_of.BeamWeight) ∩
      (∃ function_of.TargetWeight) ∩
      (∀ function_of.(BeamWeight∪TargetWeight))) ∩
   (∃ function_of.(BeamDeflectionDeviationVariable ∩
    ((∃ relativeimportance.RelativeImportance) ∩
       (∀ relativeimportance.RelativeImportance)∩
       (= relativeimportance 1)) ∩
      ((∃ function_of. BeamDeflection) ∩
      (∃ function_of. TargetDeflection) ∩
      (∀ function_of.( BeamDeflection ∪ TargetDeflection))))
  (= objective 1)
```

The DL-based representation of the cantilever beam design problem 2 enables the information associated with the cDSP to be represented in a computational means using the established vocabulary. Additionally, the advantages of DL, established in Chapters 3 and 4, can be leveraged for a specific design problem. In the following section, the cDSPs are executed and the results obtained are discussed in the context of the information explicitly represented using the formal language.

5.4 Solving the Cantilever Beam cDSPs

The primary of focus of this research is the information representation and a claim that developing a formal way to share decision making information will enable decision to be formulated more efficiently and effectively. Efficiency is very difficult to argue and in "mission-critical" situations often not the most important criteria. Additionally, formalizing the information in multiple design disciplines may have an initial overhead. However, effectiveness is related to the "correctness" of the decision based on available information. In the two cantilever beam example problems a different amount and

richness of information was captured and thus leads us to believe the effectiveness of the decision will be increased. The two cantilever beam information models were manually transformed to code and solved. In the next sections, the solution process is presented.

5.4.1 Exhaustive Search Solution Technique

The structural beam design problem is solved using an exhaustive search solution technique as a means for exploring the entire design space, eliminating the solution-dependency on specified starting values, and visualizing the design and analysis spaces for the beam problem. The exhaustive search is codified in MATLAB and results are post-processed in Microsoft Excel.

In the exhaustive search method (algorithm), a solution is determined by computing all possible solutions [160]. The exhaustive search method, or brute force method as it is sometimes called, is conceptually simple, although it is not considered to be an elegant solution approach. Unlike fine-tuned algorithms, highly-efficient that take advantage of problem characteristics, the exhaustive search method requires many calculations and is often computationally expensive. However, the exhaustive search method is not affected by ill-formed problems and therefore well-suited for many "real" engineering problems. Additionally, the exhaustive search method provides a good basis by which to judge and validate the "correctness" of results obtained with other algorithms. The solution obtained through the exhaustive search method is not affected by chosen starting values, but the accuracy of the solution is affected to the granularity of the discretization in the solution space. The exhaustive search algorithm is coded in MATLAB (see Figure 5-15).

As illustrated in Figure 5-15, the design space is defined by specifying the design variables and design parameters of interest (Step 1) in the design problem and the values of the design variable are set (Step 2). Using these design variable values, the behavior of the system is simulated using the appropriate analysis models (Step 3 & 4a-c). The results from the analysis models are returned and checked to ensure that the simulated behavior is within the design constraint and bounds (Step 5). The deviation variables and objective function is determined for all possible solutions in the design space (Step 6). The results are plotted for all solutions (Step 7). The values of the design variables are varied over he design spaces (Step 8) and the process is repeated for the entire design space. In addition, the validity of the results obtained from the analysis models are checked against the limiting assumptions of the models (Step 9). The valid results are then processed (Step 10) and plotted for visualization (Step 11).

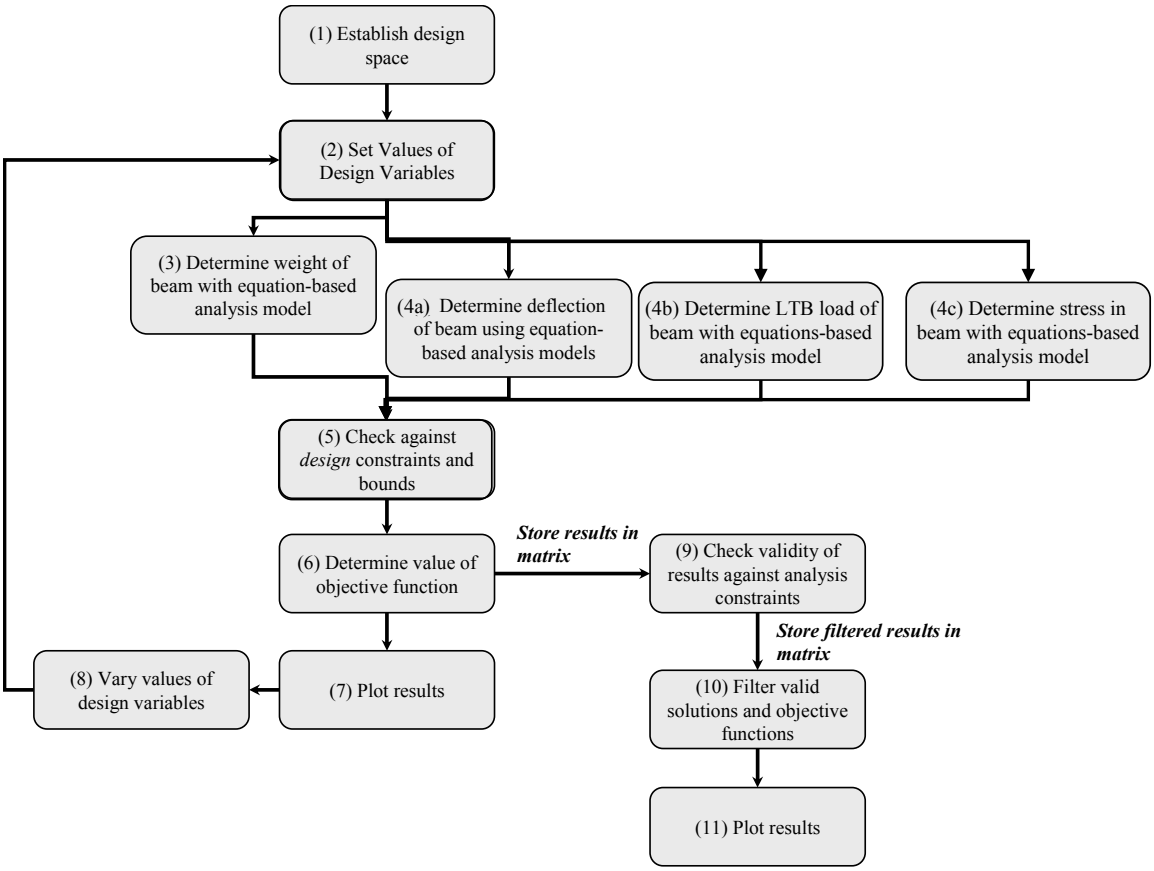


Figure 5-15: Steps in the exhaustive search solution technique for the beam design problem

In the following section, the architecture of the MATLAB code is discussed. The organization of the information models and structure of the code are organized based on the information flow and chaining of analysis models discussed in the previous sections.

5.4.2 Architecture of the Decision Problem

The implementation of the cantilever beam MATLAB code closely resembles the architecture of the DL information. This is reassuring by providing confidence in the information representation. The following is a notional schematic of the functions (see Figure 5-16).

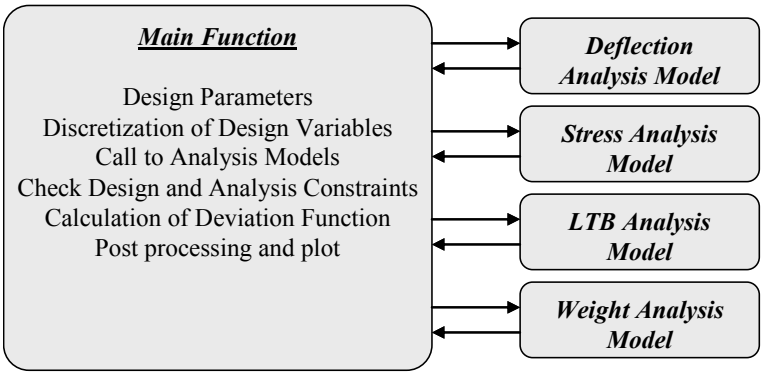


Figure 5-16: Architecture of MATLAB code

In the main function the design parameters are specified, the design variables are specified and discretized the analysis model are called and the design parameter and analysis variable are passed to the analysis models. The analysis models pass the response back to the main program where the analysis and design constraints are coded. The constraints are checked and process. The code for the cantilever beam is presented in Appendix A.

5.4.3 Cantilever Beam Decision Results

The results from the design decision are presented in Figure 5-3.

Table 5-4: Cantilever beam decision problem results

		cDSP 1			cDSP 2		
w_{weight}	$w_{deflection}$	h	b	Z	h	b	Z
0	1	0.1	0.1	0.18	0.047895	0.047895	0.95685
0.1	0.9	0.1	0.1	0.26122	0.014737	0.01	0.94704
0.2	0.8	0.1	0.1	0.34245	0.014737	0.01	0.89435
0.3	0.7	0.1	0.1	0.42367	0.014737	0.01	0.84165
0.4	0.6	0.1	0.1	0.50489	0.014737	0.01	0.78895
0.5	0.5	0.1	0.1	0.58612	0.014737	0.01	0.73626
0.6	0.4	0.01	0.01	0.53379	0.014737	0.01	0.68356
0.7	0.3	0.01	0.01	0.45611	0.014737	0.01	0.63087
0.8	0.2	0.01	0.01	0.37842	0.014737	0.01	0.57817
0.9	0.1	0.01	0.01	0.30073	0.014737	0.01	0.52547
1	0	0.01	0.01	0.22304	0.014737	0.01	0.47278

The results from the cantilever beam illustrate the importance of including analysis model meta-level constraints. For example, the solution space for cDSP1 includes the valid and invalid design space represented in Figure 5-17, whereas the solution space for cDSP2 is defined as the valid design space. The solution spaces for the two cDSPs are different because of the meta-level analysis constraints imposed by the analysis models. For example, in cDSP1 the analysis models integrated with the design decisions do not include meta-level limitations or assumptions. However, in cDSP2 the analysis models used to support the design decision include several analysis constraints. A plot of the solution space is illustrated in Figure 5-17. The total design space is decomposed into Valid and Invalid Design spaces.

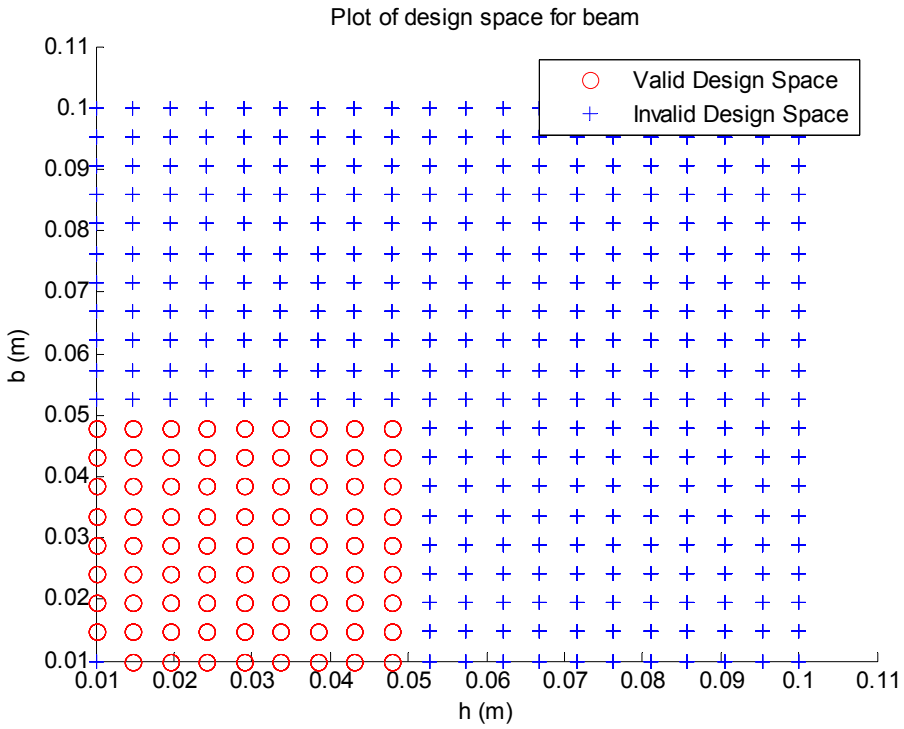


Figure 5-17: Plot of validity space for cantilever beam design problem

The red o's indicate the valid design solutions and the blue +'s represent the invalid design solutions. The cantilever beam design space is bound by upper and lower bounds on the design variables. The upper and lower bounds for the cDSP1 and cDSP2 decisions have the same values. However, in cDSP1, meta-level analysis constraints are not defined explicitly in the analysis models and thus do not constrain the solution space. Conversely in cDSP2, the design space is reduced because the analysis models used to support the design decision are restricted by several meta-level analysis constrains. The reduced design space presented in Figure 5-17 is a result of `small_deflection`, `long_beam_assumption`, and `no_lateral_torsional_buckling_occurs` meta-level analysis relationships. The valid and invalid design space is defined by the union of the constraints and bound in the design problem, thus both design and analysis constraints must be taken into consideration when solving the design decision.

The solutions to the decision are dependent on the design space, the deviation function, and the weighting values of the system goals. The solutions to the cDSPs are plotted for three different weighting scenarios in Figure 5-18 through Figure 5-20. The green circles indicate the solutions to cDSP2 and the read squares are the solutions to cDSP1.

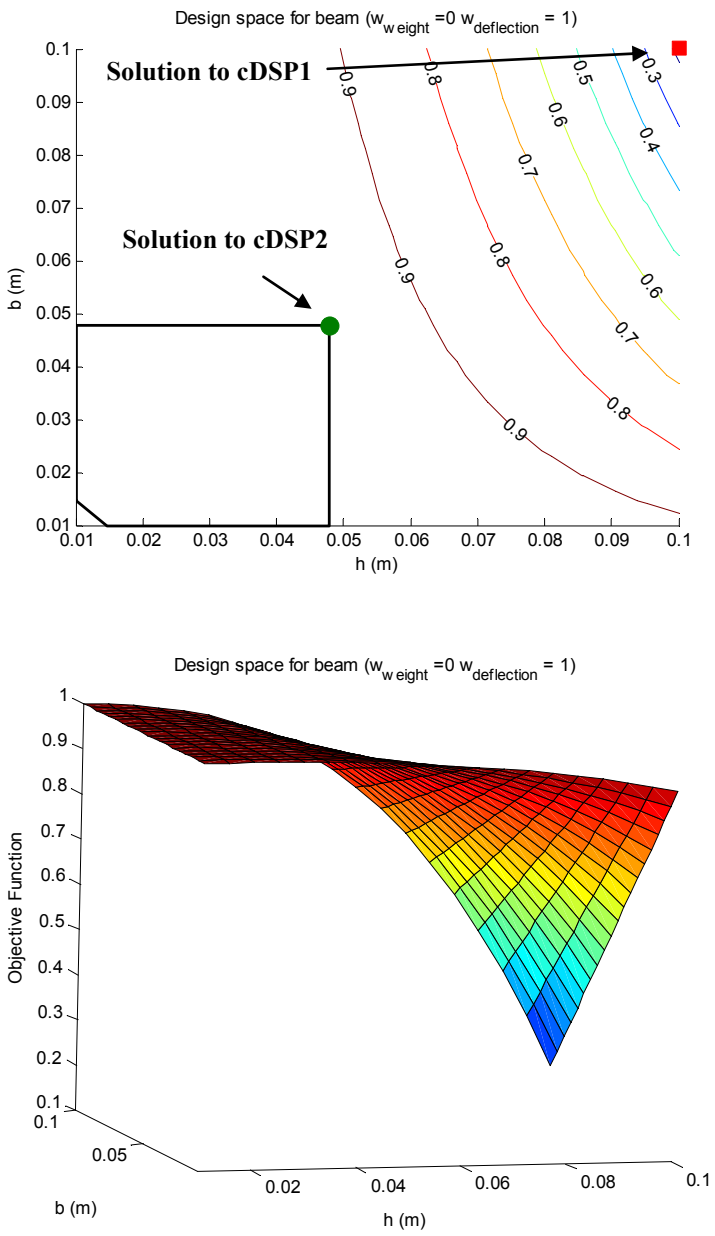


Figure 5-18: Valid and invalid solution to cantilever beam design problems (wweight = 0.0, wdeflection = 1.0)

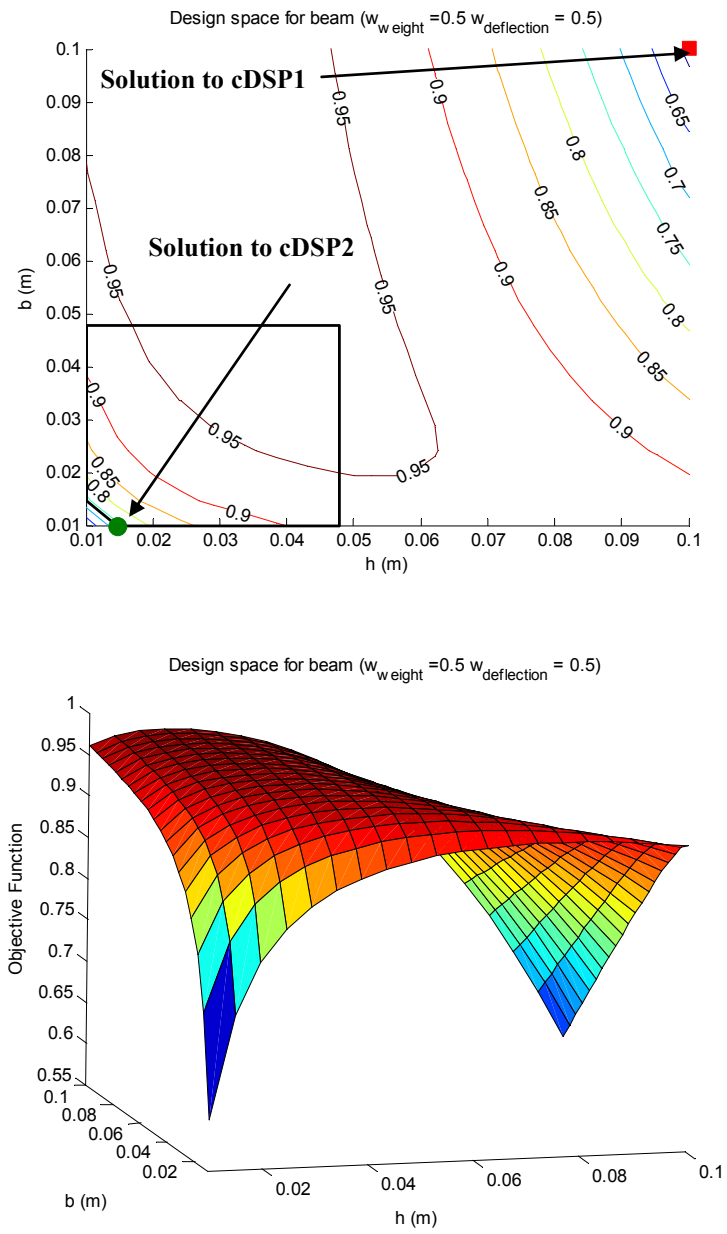


Figure 5-19: Valid and invalid solution to cantilever beam design problems (wweight = 0.5, wdeflection = 0.5)

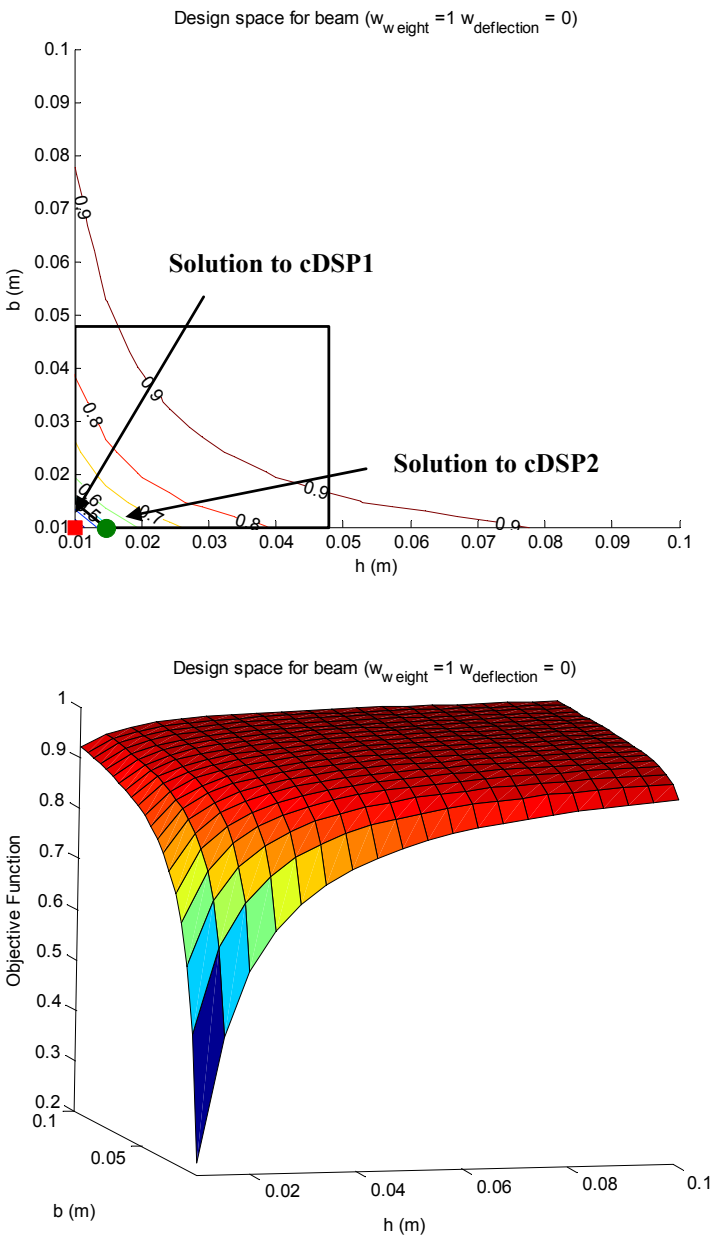


Figure 5-20: Valid and invalid solution to cantilever beam design problems (wweight = 1.0, wdeflection = 0.0)

The solutions to the cDSP1 and cDSP2 for the cantilever beam problem are different for all weighting values (see Table 5-4). The decision solutions to cDSP1 for each of the weighting scenarios have a higher performance and a lower deviation function in comparison to cDSP2. However, taking into consideration the validity of the solutions

obtained from each of the decision formulations, results in vastly different design solutions. For example, if the analysis constraints and limitations are considered, the solutions obtained from cDSP1 are invalid. In other words, the analysis models are used outside of the intended scope and thus the design solutions cannot be guaranteed. This is illustrated in the formulation of cDSP2 where the assumptions and limitations of the analysis models are explicitly modeled in the declarative and executable representations. As shown in Figures 5-18 through 5-20, the solutions obtained from cDSP2 have a higher deviation function value, but lie within the valid design space. Clearly, the *trusted* solutions perform at a lower level, but the results are valid. The execution and subsequent solutions obtained to the cantilever beam design problem demonstrate the need to capture additional information about engineering analysis models.

5.5 Discussion and Assessment of DL-Based Formal Language for the Cantilever Beam Design Problems

In the previous sections, the information associated with the design of a cantilever beam is explicitly represented using the formal language. Specifically, graphical information models are developed for representing the information networks and relationships associated with several structural analysis models and two design problems. The graphical information models are developed based on the vocabulary established in Chapter 4. These graphical models are the templates on which the computational representations are developed. Similarly, DL-based concept definitions are developed for representing analysis and decision information in a computational environment.

In this section, a critical assessment and discussion on the use of DL for developing engineering information models is presented. In Chapter 4, a general discussion on the

use of DL for developing generic decision and analysis model concepts is presented. However, the DL-based representations are not developed for specific analysis models and cDSP design decisions. Thus, the argument for DL is strengthened in this section by applying the formal language to a simple design problem. The following observations are formulated based on the cantilever beam design examples presented in this chapter:

A systematic process is followed to develop formal representations for specific cDSPs and analysis models. First, concepts that represent specific quantities are developed by establishing sub-concept definitions. For example, several concepts are defined that are subclasses to the Quantity concept. These specialized Quantity concepts are then used in conjunction with DL constructs and the cDSP vocabulary to develop complex concepts. Examples of the Quantity concepts associated with the cantilever beam design problem are illustrated in Figure 5-21.

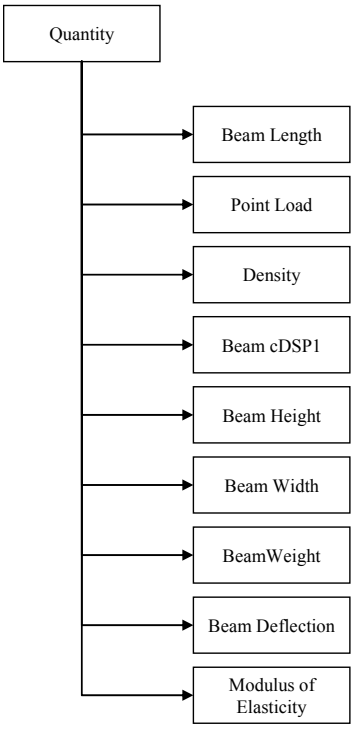


Figure 5-21: Specialized Quantity concepts associated with the cantilever beam design problem

The specialized Quantity concepts are then used to develop ConstraintRelationhsip concepts. The EquationBasedConstraintRelationship concepts are used exclusively in this research to capture the relationships between analysis and decision-related quantities. The specialized quantities are used in conjunction with the function_of property to specify definition for capturing EquationBasedConstraintRelationship concepts. Next, the constraint relationships are used to capture the information associated with several structural analysis models. The DL-based representation of the analysis models enables hierarchical taxonomies to be created based on the ConstraintRelationship concepts associated with the analysis models (see Figure 5-22).

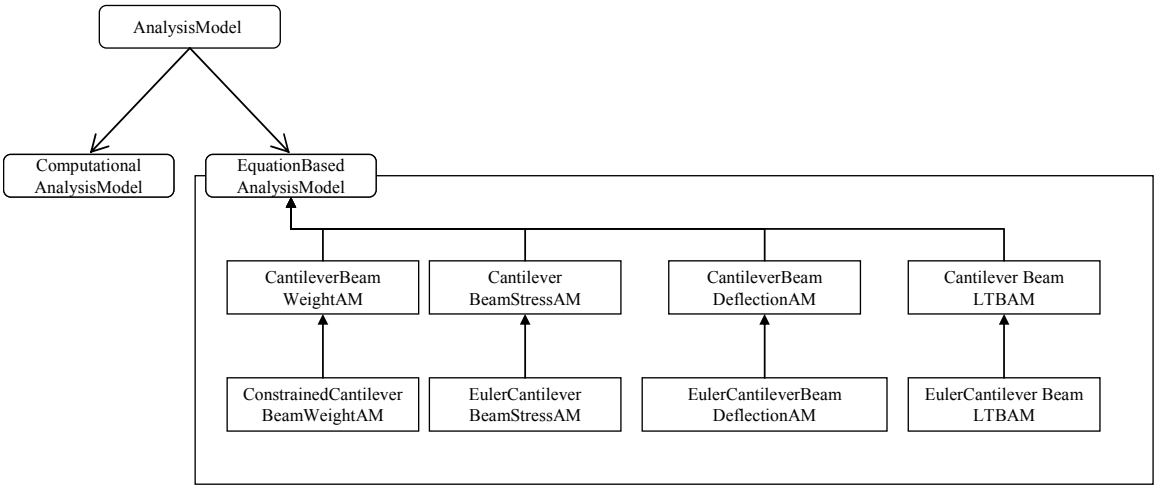


Figure 5-22: Hierarchical organization of analysis model concepts

As illustrated in Figure 5-22, the structural analysis models associated with the design of a cantilever beam are organized automatically based on the concept definitions, not on explicitly representing the subclass/superclass relationships. For example, the EulerCantileverBeamStressAM is a subclass of the CantileverBeamStressAM concept based on the concept definition and not the explicit subclass relationships specified. The relationship between the two analysis models is logically correct because the EulerCantileverBeamStressAM imposes additional restrictions on the model, thus the CantileverBeamStressAM is a more general concept definitions. In other words, the entire graph that represents the CantileverBeamStressAM concept is embedded in the graph that represents the EulerCantileverBeamStressAM concept. The consistency and organization of the analysis model hierarchical taxonomy is achieved using DL reasoning algorithms.

The specialized AnalysisModel and Quantity concepts are again used in conjunction with DL constructs and additional vocabulary to create definitions of specific design decisions (see Figure 5-23).

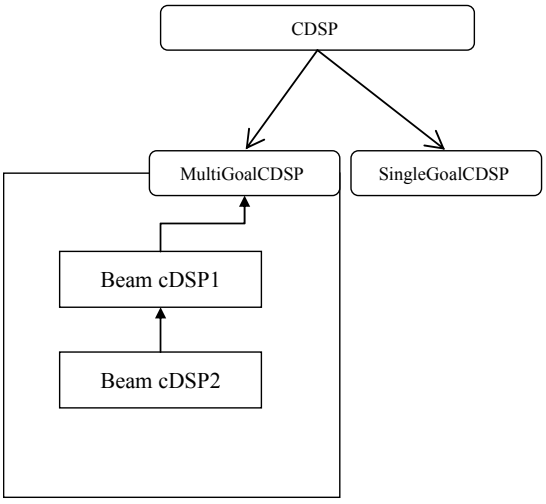


Figure 5-23: Hierarchical organization of decision model concept definitions

The organization and consistency of the design decisions are also maintained using DL reasoning algorithms. As discussed in Chapter 4, the DL reasoning algorithms provide a mathematically sound and consistent means for determining the subclass/superclass relationships based on concept definitions. Additionally, the consistency of the cDSP and AnalysisModel concepts for the cantilever beam design problem is ensured through DL reasoning. For instance, the EquationBasedConstraintRelationship concept defined in Chapter 4 is specified to take a Quantity concept in the function_of property. Thus, the function_of property must be between a ConstraintRelationship concept and a Quantity concept. If the function_of property is specified between a ConstraintRelationship concept and a different concept, the definition is considered invalid through the DL reasoner. Reuse of analysis information is illustrated through the development and subsequent integration of meta-level analysis relationships across several analysis models. For instance, elastic_stress and constantmaterialproperties meta-level constraints are applicable in several analysis models.

As discussed in Chapter 4, the strength of DL for developing engineering information is leveraging the reasoning algorithms. As illustrated in Figures 5-22 and 5-23 the cantilever beam design decisions are organized in a hierarchical manner. From a logical standpoint, this organization makes sense because it mimics the design process. For example, at the conceptual stages of design minimal information may be known about the behavior of the product. Thus, general cDSPs can be developed to determine approximate design solutions. However, as the design process progresses and more information is gained the cDSPs can become more detailed and more accurate solutions may be obtained. The organization of cDSP and analysis model concepts provides a means for traversing the hierarchy based on the *richness* of the concept assertions. In Figure 5-22, the analysis models that are lower in the hierarchy represents more specific model, whereas a concept higher in the hierarchy is a more general model. The cantilever beam design problem is summarized as:

Extensibility and robustness of the DL language - The cDSP vocabulary and DL representation have been illustrated to be extensible by (1) developing specialized Quantity concepts for capturing specific design problems, (2) developing constraint relationships for capturing the analysis and metaanalysis relationships, and (3) developing concepts that represent specific design problems. The extensibility of the language is demonstrated by developing information representation for specific analysis models and design problems by defining specialized Quantity, ConstraintRelationship, AnalysisModel, and CDSP concepts. Specifically the base concept definitions developed in Chapter 4 are applied in specific analysis domains and design problems.

Consistency and organization of the cantilever beam design information - The consistency of the cantilevered beam analysis models and design decisions is illustrated by developing a hierarchical taxonomy of engineering design decisions and analysis models based on the concept definitions. The organizational structure of the analysis model and cDSP is "up-to-date" through DL reasoning algorithms. As new analysis models are defined, the hierarchy is updated and maintained regardless of the order in which the concepts are created. DL reasoning algorithms are used to ensure the analysis models and design decisions are organized in a hierarchical manner. For example, the sub classification between structural analysis models and cDSP is based on concept definitions. The structural analysis models are automatically organized into a hierarchical taxonomy based on the meta-level assumptions and constraint associated with the model.

Integration of analysis model information - The vocabulary minimizes the ontological commitment by establishing a digital interface between analysis models and cDSP based on Quantity concepts. Information exchange is predicated on establishing a vocabulary of specialized Quantity concepts for describing the cantilever beam and associated analysis models. The cantilever beam structural analysis models are integrated into the design decisions by linking the Quantity concepts (see Figures 5-10, 5-11, 5-13, and 5-14).

The information associated with the cantilever beam design problems is successfully captured using DL. The DL-based implementation of the formal language enables 1) a computational representation of decision and analysis models to be captured, 2) the organization of decision-related information in a hierarchical taxonomy based on the

concepts definitions and not explicit relationships defined between concepts, 3) consistency of information is maintained through DL reasoning algorithms.

5.6 Verification and Validation

In this chapter, two aspects of the validation framework are addressed. Empirical structural validity (ESV), and empirical performance validity (EPV) are completed in this chapter by first evaluating the "appropriateness" of the cantilever beam example problem and then exercising the formal language (i.e., the vocabulary, information model, and DL implementation) for representing information associated with specific design decisions. In Chapter 4, the formal language is utilized for representing the general cDSP and analysis concepts. However, the information representations are not tied to specific design decisions or analysis models and thus the value of the formal language is not adequately demonstrated. Furthermore, the complexity of the information representations is not demonstrated in Chapter 4. For example, the general cDSP concept definition comprises one systemvariable.Quantity assertion, one systemparameter.Quantity assertion, and one systemgoal.Quantity assertion. These assertions correctly define the information associated with the general cDSP, but do not capture the specific system variable, parameters, and systems goals with a design decision. The examples presented in this chapter illustrate the applications and usage of the formal language for explicitly capturing the design parameter, variables, goals, constraints, and analysis models associated with the design of a cantilever beam. The aspects of verification and validation are summarized in Table 5-5.

Table 5-5: Validation and verification in Chapter 5

Empirical Structural Validation
§5.1 – The appropriateness of the cantilever beam design problem examples are first established. The detailed test plan and purpose of the examples are summarized in Table 5-1. The purpose of the cantilever beam example problem is established in the context of answering the research questions and hypotheses. The cantilever beam design problem examples are appropriate because they enable the formal language to be demonstrated for explicitly capturing the information associated with multi-disciplinary design decisions and disciplinary analysis models. Additionally, the extensibility and robustness of the language is demonstrated. Finally, the examples presented in this chapter provide a means for organizing and checking the consistency of the modeling concepts.
Empirical Performance Validation
§5.2 – The formal language is utilized for explicitly capturing the information associated with multi-disciplinary analysis models. Analysis models for beam weight, stress, deflection, and buckling are represented in a computational means using the formal language. The analysis models are published to a repository and hierarchically organized. The quantities and constraints associated with the analysis models are explicitly represented, thus enabling the analysis models to be plugged into engineering design decisions. §5.3 – The information associated with two cDSPs is represented using the formal language. The design decisions are closely related, the first do not take into account analysis constraints while the second cDSP does. §5.4 – The design decisions represented in Section 5.3 and the analysis models in 5.2 are manually translated to executable decision representation. The results from the decisions are discussed and compared to illustrate the importance of capturing information associated with design decisions and analysis models in a computational means.

Empirical structural validation (ESV) involves accepting the appropriateness of the example problems used to verify the performance of the method. The example discussed in this chapter is a multi-disciplinary design problem. The design problems involve several analysis models that are used to predict engineering behaviors and phenomenon from several disciplines. The analysis models are coupled together through several mechanisms including shared design variables and parameters and through chaining of analysis quantities. The design problems are constrained by bounds on the system variables, design and behavioral requirements, and constraints imposed by the analysis models. For example, the analysis models used to predict the structural performance of the beam within a set of underlying assumptions and limitations. Additionally, the design problems require the integration and exchange of decision-related information from multiple design perspectives. We believe the example problem is of reasonable complexity and enables several different aspects of the research questions to be addressed. The design problem enables the effect of analysis constraints to be explored on the outcome of the design decision. Additionally, the information associated with the analysis models presented in Section 5.2 is represented and integrated into decision models in Section 5.3. Hence, the cantilever beam design problem is appropriate for the validation of the formal language.

Empirical performance validation (EPV) consists of accepting the usefulness of the outcome with respect to the initial purpose and accepting that the achieved usefulness is related to applying the method. The empirical performance validation in this chapter is carried out by explicitly representing the information associated with disciplinary analysis models in Section 5.2, using the formal language to capture the information

associated with engineering design decision and integrate analysis models into the decisions in Section 5.3, and execute the design decision in order to gain a better understanding of the effect of representing and exchanging information in design decisions in Section 5.4

The formal language is shown to be effective for representing decision-related information in a computational means. The use of the formal language for representing analysis model information is shown in Section 5.2. It is shown that the formal language enable the quantities associated with a model to be unambiguously captured and the constraints associated with analysis models to be represented. Graphical representations of the analysis information provide a means for visualizing the information network. Additionally, the formal language is used for explicitly representing decision information and providing a means for integrating analysis model knowledge into engineering design decisions. In Section 5.3 two design decisions are represented using the formal language that involves disciplinary analysis models. The formal language provides a structured, computationally-based approach for capturing and exchanging decision-related information. The design decisions are then "translated" to a procedural representation and executed. Based on the results, it is argued that formal language is appropriate for explicitly capturing and exchanging the information associated with multi-disciplinary design problems.

5.7 Chapter Synopsis

In this chapter the information model and DL language are used for explicitly capturing the information associated with a cantilever beam design problems (see Figure 5-24).

- ✓ To demonstrate the use of the formal language for a simple, single-disciplinary design problem – The information associated with two cantilever beam design problems is explicitly represented using the formal language. The semantics of two cDSPs are represented including design requirements, four analysis models, three design goals, design variables, and design parameters are captured.
- ✓ To capture the information structure associated with disciplinary analysis models – The information associated with four analysis models, including analysis relationships and meta-relationships, are captured.

This chapter illustrates how the language is used for describing disciplinary analysis models and how the analysis models are inter-related. The language is used to describe structural decisions and analysis models. Additionally, the importance of capturing the assumptions and limitations of analysis models is illustrated by executing a design decision and showing the effects on the solution results. In the next chapter, the language is used for explicitly representing multi-disciplinary design decisions and analysis models.

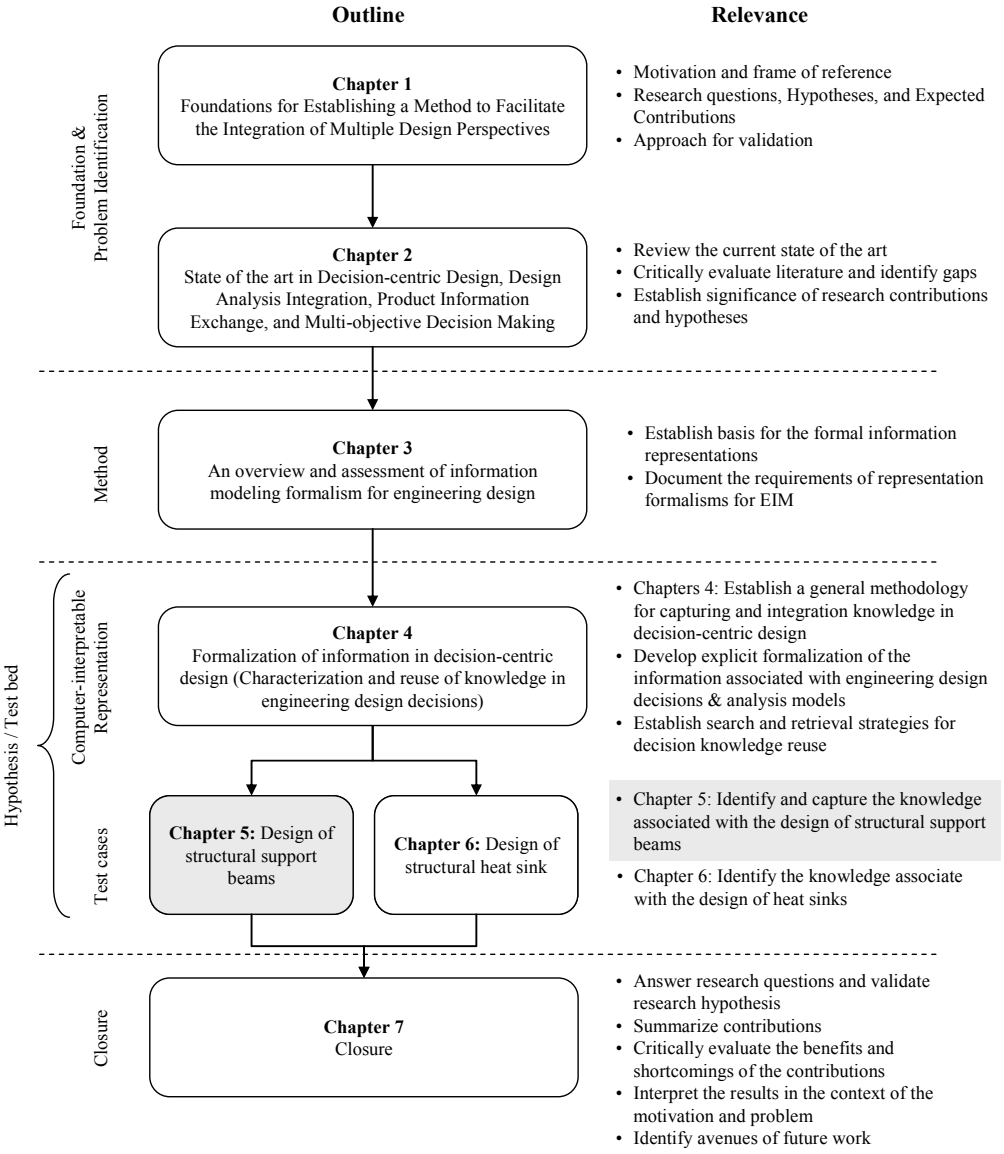


Figure 5-24: Outline of dissertation

CHAPTER 6:

DESIGN OF FIN ARRAY HEAT SINKS

Chapter Aims

- To demonstrate the use of the formal language for a complex, multi-disciplinary design problem
- To explicitly capture the relationships, both analysis and meta-level, for disciplinary analysis models
- To illustrate the importance of capturing design information to enable integration and exchange

In the fin array heat sink described in this chapter, the information model and knowledge representation are exercised for capturing information associated with multi-disciplinary analysis models and multi-objective design decisions. The information model facilitates capturing and reusing decision making information.

6.1 Problem Overview: Design of Structural Fin Array Heat Sink

Heat generation is a natural, and often undesirable, side-effect that affects the performance, reliability and life expectancy of the system of electronic equipment [155]. Significant internal heat is generated within semiconductor devices that affect the lifecycle performance the entire electronic system. Thus, thermal management in modern electronic devices is required to eliminate catastrophic failure and fulfill overall performance, reliability, and life expectancy requirements [155]. A common example of thermal management is cooling the central processing unit (CPU) in the personal computer (PC). Heat can be dissipated from the CPU to the environment through three

modes: conduction, convection, and radiation. Conductive heat transfer deals with the flow of heat in solid bodies. Convective heat transfer deals with heat exchange between a solid surface and a fluid. Radiative heat transfer occurs between two surfaces of different temperatures [86]. However, the effectiveness of these heat transfer modes varies. Fin array heat sinks are commonly used to enhance thermal dissipation from the CPU to the surrounding environment through conduction and convection modes of heat transfer. Examples of finned array heat sinks are presented in Figure 6-1.

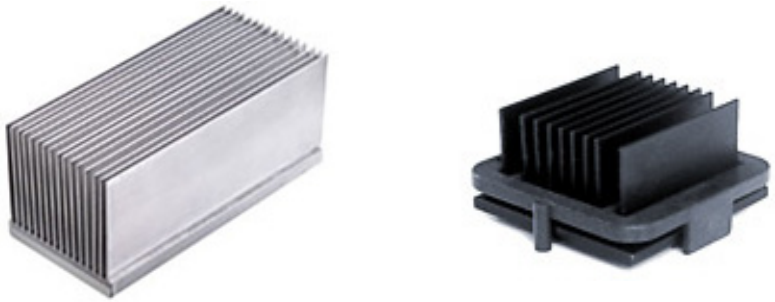


Figure 6-1: Examples of finned heat sinks for electronic chip packages (Source: left image: [8], right image [119])

Fin array heat sinks, by themselves, are passive devices that dissipate heat energy from a hot surface to a cooler surface or surrounding environment through natural convection. They are manufactured from materials with high thermal conductivities to draw heat away from the heat generation sources and dissipate the heat to the air through convection. Aluminum and copper are commonly used because of the high thermal conductivity. However, to further enhance heat dissipation, cooling fans are commonly added to increase the airflow, thereby increasing the convective coefficient. A sample structural heat sink is illustrated in Figure 6-2.

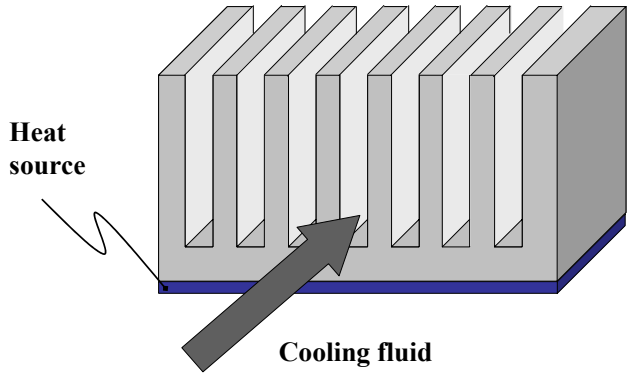


Figure 6-2: Forced convective cooling air for increased heat dissipation

The design of fin array heat sinks is a multi-disciplinary design activity that often involves tradeoffs between thermal, structural objectives, and several design requirements. For this example, the information model and DL ontology are utilized to explicitly capture the information associated with multi-objective decision-making with multi-disciplinary analysis models for structural and heat transfer. The focus is to demonstrate the extensibility and robustness of the information models for facilitating multifunctional materials design in a deterministic, parametric design context. In this example, the information model is demonstrated to be an effective model for composing and formulating engineering design decisions, capturing limitations of engineering analysis models, exploring the effects of different model formulations, and capturing the effects on decision results.

The fin array heat sink examples are intended to demonstrate how the information model and DL-ontology can be used for (1) explicitly representing analysis model knowledge, (2) capturing limitations and assumptions of engineering analysis models (3) capturing the computational assumption associated with computer simulations, (4) modeling multi-disciplinary engineering design decisions, and (5) reusing decision-

related knowledge in the decision making process. A summary of the test plan and validation examples is presented in Table 6-1.

Table 6-1: Test plan and outline

Step 1:	Identify the analysis support models for designing the fin array heat sink. Derive the analysis relationships, assumptions, and computational limitations of the analysis models.
Step 2:	Capture the analysis model knowledge using the conceptual information model
Step 3:	Represent the analysis model knowledge computationally using the DL ontology
***Purpose*:**	Demonstrate the use of the conceptual information model and DL representation for explicitly capturing multi-disciplinary analysis models. Demonstrate that the formal language is extensible and robust, in that is can be used to model analysis information from multiple disciplines for a complex design example.
Step 5:	Formulate several multi-objective heat sink design problems that involve a variety of system goals, constraints and analysis models.
Step 6:	Represent the decision-related knowledge using the conceptual information model.
Step 7:	Implement the heat sink design decisions using the DL ontology.
Step 8:	Use standard reasoning algorithms to organize, check for consistency, and retrieve engineering design decisions.
***Purpose*:**	Demonstrate the use of the conceptual information model and DL representation for explicitly representing multi-objective design decisions. Demonstrate that the formal language is extensible and robust, in that is can be used to model analysis information from multiple disciplines.

6.2 Information Modeling of Heat Sink Analysis Models

The engineering analysis models are presented in accordance with the information model and DL language defined in Chapter 4.

6.2.1 Thermal and Fluid Mechanics Analysis Models

In this section, the information associated with thermal and fluid analysis models is explicitly represented. The analysis models presented in this section are used for determining behaviors related to heat transfer, temperature, thermal resistance, and fluid flow phenomena. The coupling of thermal and fluid phenomena is commonplace and often referred to as conjugate analysis. Thus, while heat transfer and fluid mechanics are considered separate engineering discipline, in this research they are presented together because of the close relationship.

6.2.1.1 Finned Array Heat Thermal Resistance Analysis Model

Fin array heat sinks are comprised of several *extended surfaces.* Extended surfaces are solid bodies that transfer energy by conduction within the body and by convection between the body and a fluid. The total heat transfer in a fin array heat sink is determined by first establishing the governing relationships for a single extended surface. The heat transfer in a fin is assumed to be one-dimensional, along the length of the fin. A single rectangular fin, illustrated in Figure 6-3, is exposed to a cooling fluid with a convective heat transfer coefficient, *h.*

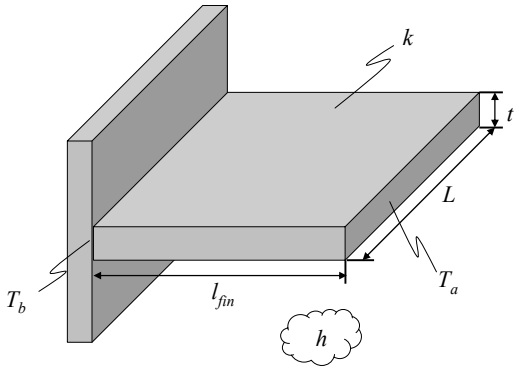


Figure 6-3: Single rectangular fin

The thermal conductivity of the fin material is strongly related to the temperature distribution in the fin. Therefore, the fin should be made of a highly conductive material to minimize temperature gradients from the base of the fin to the tip of the fin. Ideally, a material with an infinitely large thermal conductivity would eliminate the temperature gradient along the fin, resulting in a uniform temperature in the fin equal to the base temperature and therefore a maximum heat flow from the extended surface to the surrounding. Practically, a temperature gradient exists along the length of the fin that reduces the total heat transfer rate. A rigorous analysis is presented in [155], resulting in the following equation for computing heat transfer in a single fin:

$$q = \eta_f h A_f \left(T_b - T_a\right) \qquad \textbf{6.1}$$

$$A_f = 2 \cdot L \cdot L_c \qquad \textbf{6.2}$$

$$L_c = l_{fin} + \frac{t}{2} \qquad \textbf{6.3}$$

where $h\left(W/m^2\,K\right)$ is the convective coefficient, A_f is the base area of the fin, η is the fin efficiency, l_{fin} is the length of the fin, t is the thickness of the fin, and T_a and T_b are

the temperatures in the fin and at the fin base respectively. Fin efficiency is the ratio between the maximum heat transfer based on a uniform temperature assumption and the actual heat transfer. For a straight fin with a uniform cross section and adiabatic tip conduction, fin efficiency is given as:

$$\eta_f = \frac{\tanh\left(m \cdot L_c\right)}{m \cdot L_c} \qquad \textbf{6.4}$$

where L_c is the corrected length of the fin.

$$L_c = L + \frac{t}{2} \qquad \textbf{6.5}$$

where L is fin length and t is the thickness of the fin. If the width of the fin (L), is much greater than the thickness (t), the perimeter of the fin can be approximated as:

$$P = 2L \qquad \textbf{6.6}$$

Thus, the corrected profile area of the fin is given as $A_p = L_c t$ and m is given as:

$$m = \sqrt{\frac{2h}{kA_p}} \cdot L_c^{3/2} \qquad \textbf{6.7}$$

where h is the convective coefficient, k is the thermal conductivity of the fin material, and A_p is the corrected fin cross sectional area. The relationships given in Equation 6.7 are valid with the width of the fin is much greater than the thickness of the fin. A detailed derivation is including in [66]. The equation for computing heat transfer in a single fin is based on the assumption of adiabatic tip condition ($dT/dx\big|_{x=L} = 0$).

It is shown in [66] that a correction factor can be used if the tip condition is not actually adiabatic.

$$\eta_f = \frac{\tanh\left(mL_c\right)}{mL_c} \qquad \textbf{6.8}$$

The errors associated with Equations 6.5 and 6.8 are negligible if $ht/k \le 0.0625$ for rectangular fins. The analytical relationships established for a single fin serve as the basis for fin array heat sinks commonly used to cool electronic components. An array of thermal fins is presented in Figure 6-4.

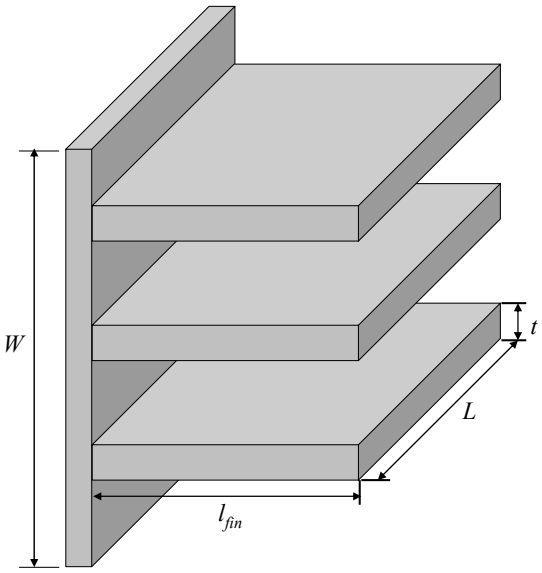


Figure 6-4: Array of rectangular fins

The thermal efficiency of an array of fins attached to a base is given as:

$$\eta_o = \frac{q_t}{hA_t\theta_b} \qquad \textbf{6.9}$$

where q_t is the total heat transfer, A_t is the total area for the fins and the exposed portion of the base. For rectangular fin array, the total area of the fins is given as:

$$A_t = NA_f + A_b \quad \textbf{6.10}$$

where N fins is the number of fins that make up the array, A_b is the prime surface of the heat sink, and A_f is the area of a single fin given as:

$$A_f = 2 \cdot L \cdot L_c \quad \textbf{6.11}$$

$$A_b = W - \left(N \cdot t\right) \quad \textbf{6.12}$$

The total convective heat transfer from the exposed base surface and the fins is given as:

$$q_t = N\eta hA_f\theta_b + hA_b\theta_b \quad \textbf{6.13}$$

Substituting Equations 6.10, 6.13, and 6.9 results in the thermal efficiency of a fin array heat sink to be computed as:

$$\eta_0 = 1 - \frac{NA_f}{A_t}\left(1 - \eta_f\right) \quad \textbf{6.14}$$

The thermal resistance of the fin array is given as:

$$R_t = \frac{\theta_b}{q_t} \quad \textbf{6.15}$$

Thus, Equations 6.9, 6.14 and 6.15 results in:

$$R_t = \frac{1}{\eta_0 \bar{h} A_t} \quad \textbf{6.16}$$

The thermal resistance model relationship and assumptions are represented using the graphical information model and DL ontology. The analysis model is illustrated

graphically in Figure 6-5. The information network captures the relationships between the analysis quantities, the analysis relationships and meta-level relationships. The graphical representation captures the dependence on *external* quantities required for using the model. While the quantities are not captured in detail in Figure 6-5 for clarity, the illustration indicates that additional information must be instantiated to enable the model to be executed. It is shown in Figure 6-5, that the thermal resistance analysis model has nine quantities directly related to the analysis model.

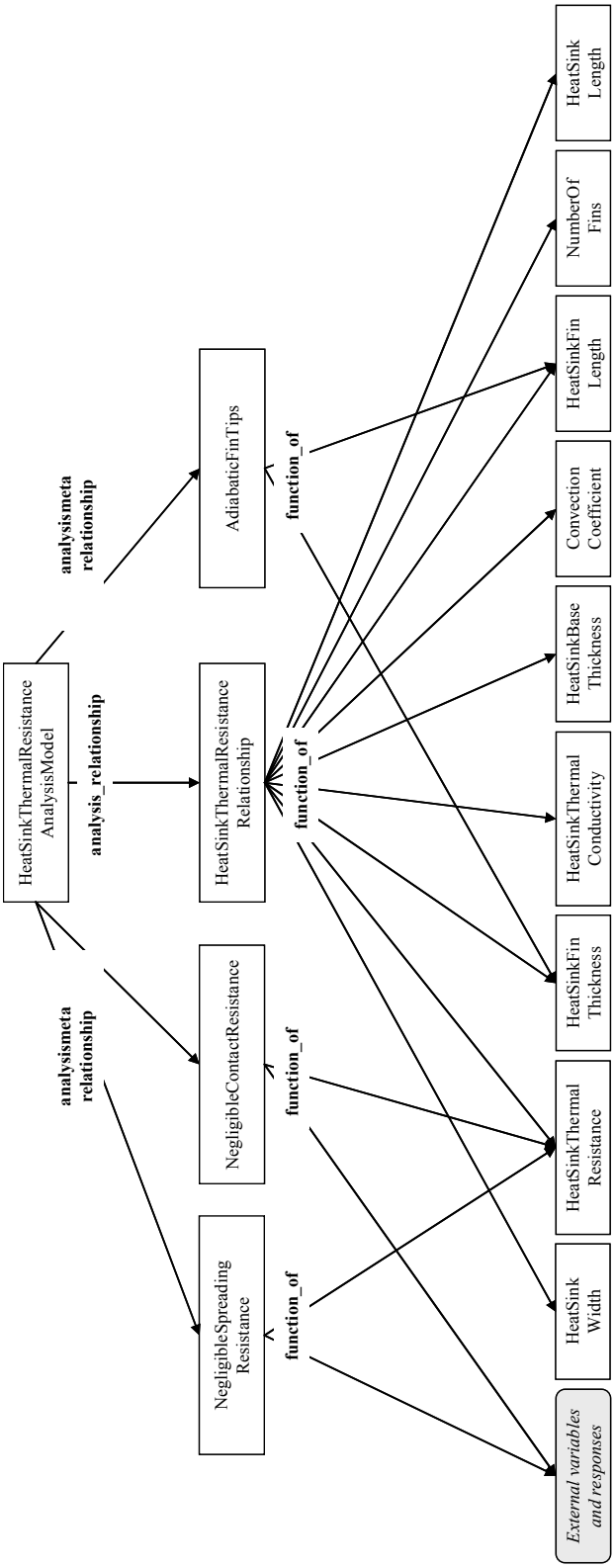


Figure 6-5: Graphical representation of thermal resistance analysis model

The thermal resistance analysis model concept is specified using DL and the cDSP vocabulary as:

```
Class(HeatSinkThermalResistanceAnalysisModel complete
and(
   (∃ analysisrelationship.HeatSinkThermalResistanceRelationship)
   (∀ analysisrelationship.HeatSinkThermalResistanceRelationship)
   (∃ analysismetarelationship.NegligibleContactResistance)
   (∃ analysismetarelationship.NegligibleSpreadingResistance)
   (∃ analysismetarelationship.AdiabaticFinTips)
   (∀ analysismetarelationship.(AdiabaticFinTips ∪
   NegligibleSpreadingResistance ∪ NegligibleContactResistance)
SubClassOf(HeatSinkThermalResistanceAnalysisModel
   EquationBasedAnalysisModel)))
```

```
Class(HeatSinkThermalResistanceRelationship partial
and(
   EquationBasedConstraintRelationship ∩
   (∃function_of_quantity.HeatSinkWidth) ∩
   (∃function_of_quantity.HeatSinkThermalResistance) ∩
   (∃function_of_quantity.HeatSinkFinThickness) ∩
   (∃function_of_quantity.HeatSinkThermalConductivity) ∩
   (∃function_of_quantity.HeatSinkBaseThickness) ∩
   (∃function_of_quantity.ConvectionCoefficient) ∩
   (∃function_of_quantity.HeatSinkFinLength) ∩
   (∃function_of_quantity.NumberOfFins) ∩
   (∃function_of_quantity.HeatSinkLength) ∩
   (∀function_of_quantity.(ConvectionCoefficient∪ HeatSinkBaseThickness
   ∪ HeatSinkFinLength ∪ HeatSinkFinThickness ∪ HeatSinkLength ∪
   HeatSinkThermalConductivity ∪ HeatSinkThermalResistance ∪
   HeatSinkWidth ∪ NumberOfFins))))
```

```
Class(NegligibleContactResistance complete
and(
   EquationBasedConstraintRelationship ∩
   (∃function_of_quantity.HeatSinkThermalResistance) ∩
   (∃function_of_quantity.HeatSinkContactResistance) ∩
   (∀function_of_quantity.(HeatSinkContactResistance ∪
   HeatSinkThermalResistance))))
```

```
Class(NegligibleSpreadingResistance complete
and(
   EquationBasedConstraintRelationship ∩
   (∃function_of_quantity.HeatSinkThermalResistance) ∩
   (∃function_of_quantity.HeatSinkSpreadingResistance) ∩
   (∀function_of_quantity.(HeatSinkSpreadingResistance ∪
   HeatSinkThermalResistance))))
```

```
Class(AdiabaticFinTips complete
and(
   EquationBasedConstraintRelationship ∩
   (∃function_of_quantity.HeatSinkFinThickness) ∩
   (∃function_of_quantity.HeatSinkFinLength) ∩
   (∀function_of_quantity.(HeatSinkFinLength))))
```

6.2.1.2 Finned Array Heat Sink Temperature Analysis Model

The temperature is in the chip is computed using a thermal resistance circuit model (see Figure 6-6).

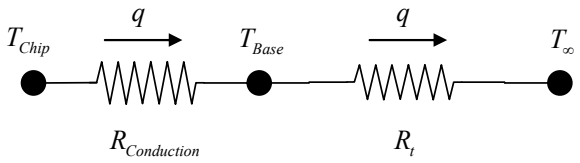


Figure 6-6: Thermal circuit illustration for electronic chip assembly

The thermal resistance of the fin array is given in Equation 6.16. The conductive thermal resistance of the heat sink base is given as:

$$R_{Base\ Conduction} = \frac{t_{\text{Base}}}{k_{\text{Heat Sink}} \cdot L \cdot W} \qquad \textbf{6.17}$$

where t_{Base} is the thickness of the heat sink base. The total thermal resistance of the fin array heat sink is

$$R_{Total} = R_{BaseConduction} + R_t \quad \textbf{6.18}$$

The temperature in the chip is computed using Equations 6.18 and 6.19:

$$q_{Total} = \frac{\Delta T_{Chip-\infty}}{R_{Total}} \quad \textbf{6.19}$$

$$\Delta T_{Chip-\infty} = T_{Chip} - T_{\infty} \quad \textbf{6.20}$$

$$T_{Chip} = R_{Total} \cdot q_t + T_{\infty} \quad \textbf{6.21}$$

Equation 6.21 is valid based on the assumption that 100% of the chip power is dissipated through the fin array heat sink. The analysis model for computing the temperature in the heat sink is represented graphically in Figure 6-7.

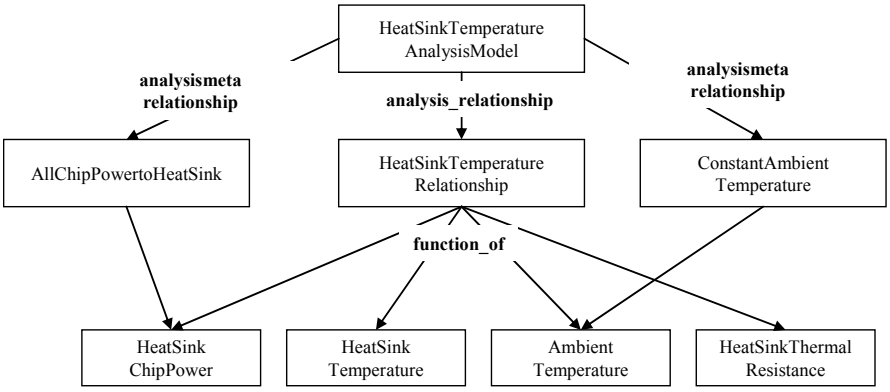


Figure 6-7: Graphical representation of heat sink temperature analysis model

As illustrated in Figure 6-7, the heat sink temperature model is characterized by two meta-level relationships, a single analysis relationship, and four analysis quantities. The heat sink temperature analysis model is representing using DL in as:

```
Class(HeatSinkTemperatureAnalysisModel complete
and(
   (∃ analysisrelationship.HeatSinkTemperatureRelationship)
   (∀ analysisrelationship.HeatSinkTemperatureRelationship)
   (∃ analysismetarelationship.ConstantAmbientTemperature)
   (∃ analysismetarelationship.AllChipPowertoHeatSink)
   (∀ analysismetarelationship.(AllChipPowertoHeatSink ∪
   ConstantAmbientTemperature))

SubClassOf(HeatSinkTemperatureAnalysisModel
   EquationBasedAnalysisModel)))
```

```
Class(HeatSinkTemperatureRelationship partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.AmbientTemperature)
   (∃ function_of_quantity.HeatSinkTemperature)
   (∃ function_of_quantity.HeatSinkThermalResistance)
   (∃ function_of_quantity.HeatSinkChipPower)
   (∀ function_of_quantity.(AmbientTemperature∪HeatSinkChipPower ∪
   HeatSinkTemperature∪HeatSinkThermalResistance))))
```

```
Class(ConstantAmbientTemperature partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.AmbientTemperature)
   (∀ function_of_quantity.AmbientTemperature)))
```

```
Class(AllChipPowertoHeatSink partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.HeatSinkChipPower)
   (∀ function_of_quantity.HeatSinkChipPower)))
```

6.2.1.3 Fluid Velocity Analysis Model

To compute the convective heat transfer coefficient, the velocity of the cooling fluid must be calculated. The velocity of thee cooling fluid is:

$$v = \frac{\dot{V}}{A_{channel,total}} \qquad \textbf{6.22}$$

where $A_{channel}$ is the cross section area of the channel and $\dot{V}$ is the volumetric flow rate of the cooling fluid in the fin array heat sink.

$$A_{channel,total} = \left(W - N \cdot t\right) \cdot l_{fin} \qquad \textbf{6.23}$$

The velocity equation is based on the assumptions that there is no by-pass flow of the air around the heat sink, the velocity in each of the channels is uniform, and there is negligible pressure in the channels. The fluid velocity analysis model is illustrated graphically in Figure 6-8.

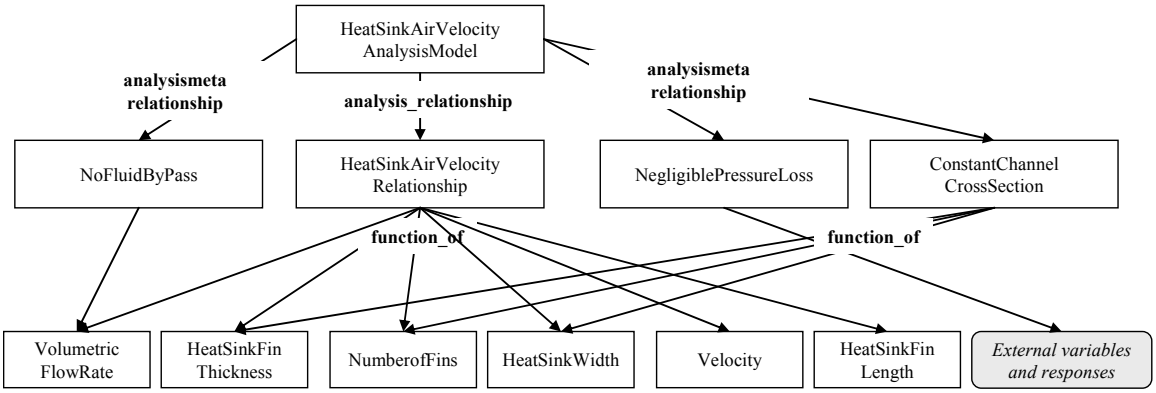


Figure 6-8: Graphical representation of heat sink fluid velocity analysis model

The digital interface of the `HeatSinkAirVelocityAnalysisModel` is defined as the aggregation of: `HeatSinkFinLength`, `HeatSinkFinThickness`, `HeatSinkWidth`, `NumberOfFins`, `Velocity`, and `VolumetricFlowRate`. The analysis model is constraint by three meta-level analysis relationships:

- There is no cooling fluid by-pass – 100% of the cooling fluid flow through the channels in the heat sink

- NegligiblePressureLoss – the cooling fluid does not experience a high pressure loss and therefore is of uniform velocity along the length of the heat sink
- ConstantCrossSection – the cross section in which the cooling fluid flow does not change along the length of the beam.

The analysis model is implemented using DL:

```
Class(HeatSinkAirVelocityAnalysisModel complete
and(
   (∃ analysisrelationship.HeatSinkFluidVelocityRelationship)
   (∀ analysisrelationship.HeatSinkFluidVelocityRelationship)
   (∃ analysismetarelationship.NegligiblePressureLoss)
   (∃ analysismetarelationship.NoFluidBypass)
   (∀ analysismetarelationship.(NegligiblePressureLoss ∪
   NoFluidBypass))
SubClassOf(HeatSinkAirVelocityAnalysisModel
   EquationBasedAnalysisModel)))
```

```
Class(HeatSinkFluidVelocityRelationship partial
and(
   EquationBasedAnalysisRelationship
   (∃ function_of_quantity.HeatSinkFinLength)
   (∃ function_of_quantity.VolumetricFlowRate)
   (∃ function_of_quantity.Velocity)
   (∃ function_of_quantity.NumberOfFins)
   (∃ function_of_quantity.HeatSinkWidth))
   (∃ function_of_quantity.HeatSinkFinThickness)))
   (∀ function_of_quantity.(HeatSinkFinLength∪ HeatSinkFinThickness ∪
   HeatSinkWidth∪NumberOfFins∪Velocity ∪ VolumetricFlowRate))))
```

```
Class(NoFluidBypass partial
and(
   EquationBasedAnalysisRelationship
   (∃ function_of_quantity.VolumetricFlowRate)
   (∀ function_of_quantity.VolumetricFlowRate)))
```

```
Class(NegligiblePressureLoss partial
and(
   EquationBasedAnalysisRelationship
   (∃ function_of_quantity.ReynoldsNumber)
   (∀ function_of_quantity.ReynoldsNumber)))
```

```
Class(ConstantChannelCrossSection partial
and(
   EquationBasedAnalysisRelationship
   (∃ function_of_quantity.HeatSinkWidth)
   (∃ function_of_quantity.NumberofFins)
   (∃ function_of_quantity.HeatSinkThickness)
   (∀ function_of_quantity.(HeatSinkFinThickness ∪
   HeatSinkWidth∪NumberOfFins))))
```

6.2.1.4 Hydraulic Diameter Relationship

There are many well known published correlations for heat transfer in circular tubes. However, many engineering application, including fin array heat sink rely on noncircular channels. An effective diameter, called the *hydraulic diameter,* is often used as an approximation for computing fluid flow and heat transfer in noncircular channels. The hydraulic diameter is computed as:

$$D_h = \frac{4A_c}{P} \qquad \textbf{6.24}$$

where P is the perimeter of the channel and A_c is the cross section of each channel. For rectangular channels in the heat sink they are computed as:

$$A_c = \frac{A_{channel}}{N-1} \qquad \textbf{6.25}$$

$$P = 2\left(l_{fin} + \frac{(W - Nt)}{N-1}\right) \qquad \textbf{6.26}$$

The hydraulic diameter must be used to compute the convective coefficient for noncircular channels with turbulent flow. The Reynolds number for hydraulic diameters is computed in the next section. The analysis model for computing hydraulic diameter is illustrated in Figure 6-9.

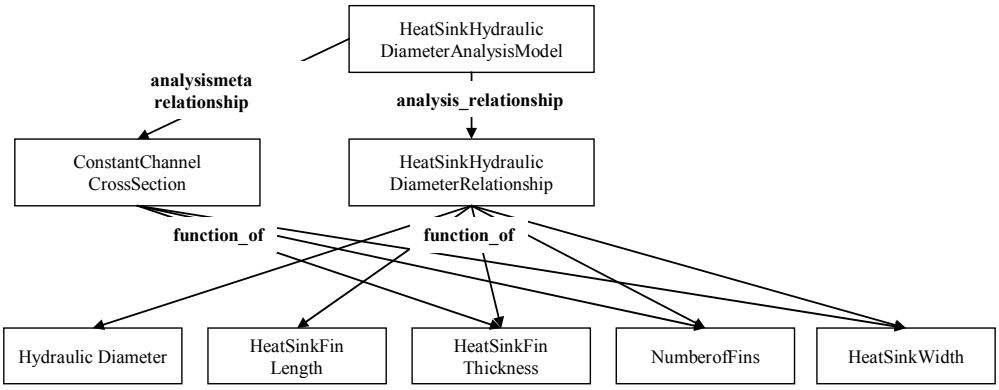


Figure 6-9: Graphical representation of hydraulic diameter analysis model

The digital interface of the `HeatSinkHydraulicDiameterAnalysisModel` is defined as the aggregation of: `HeatSinkFinLength`, `HeatSinkFinThickness`, `HeatSinkWidth`, `NumberOfFins`, and `HydraulicDiameter`. The analysis model is constraint by the `ConstantCrossSection` meta-level analysis relationship. The `ConstantCrossSection` constraint is reused from a previously defined analysis model. The analysis model is implemented using DL:

```
Class(HeatSinkHydraulicDiameterAnalysisModel complete
and(
    (∃ analysisrelationship.HydraulicDiameterRelationship)
    (∀ analysisrelationship.HydraulicDiameterRelationship)
    (∃ analysismetarelationship.ConstantChannelCrossSection)
    (∀ analysismetarelationship.ConstantChannelCrossSection)

SubClassOf(HeatSinkHydraulicDiameterAnalysisModel
EquationBasedAnalysisModel)
```

```
Class(HydraulicDiameterRelationship partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.HeatSinkFinLength)
  (∃ function_of_quantity.HeatSinkFinThickness)
  (∃ function_of_quantity.NumberOfFins)
  (∃ function_of_quantity.HydraulicDiameter)
  (∃ function_of_quantity.HeatSinkWidth)
  (∀ function_of_quantity.(HeatSinkFinLength∪HeatSinkFinThickness∪
  HeatSinkWidth∪HydraulicDiameter∪NumberOfFins))
```

6.2.1.5 Reynolds Number Analysis Model

Reynolds number is used to determine the flow regime of the fluid and is computed as:

$$\mathrm{Re}_{D_h} = \frac{\rho v D_h}{\mu} \qquad \textbf{6.27}$$

For internal flow in a duct, laminar flow is considered if $\mathrm{Re}_{D_h} < 2300$, turbulent flow is greater than 2300. Reynolds number is computed, because the correlation for determining the convective heat transfer coefficient assumes fully developed laminar flow. The Reynolds Number analysis model is illustrated graphically in Figure 6-10.

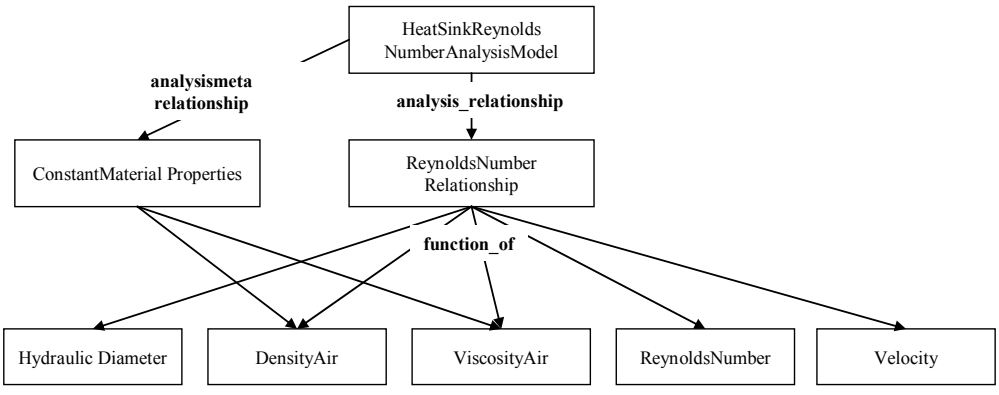


Figure 6-10: Graphical representation of Reynolds number analysis model

The digital interface of the HeatSinkReynoldsNumberAnalysisModel is defined as: DensityAir, ViscosityAir, HydraulicDiamter, Velocity, and ReynoldsNumber. The Reynolds Number analysis model required coupling to external analysis models. For example, to compute Reynolds Number, the velocity of the fluid moving in the channel must be determined, and the hydraulic diameter of the channel must be computed. Thus, to enable the computation of Reynolds Number, the analysis model must be coupled to external analysis models. Additionally, the Reynolds Number analysis model is constraint by meta-level analysis relationships, ConstantMaterialProperties, which specifies the cooling fluid must have material properties, such as the density and viscosity of the fluid, to be constant. A DL-based representation is implemented using the graphical representation and the cDSP vocabulary as:

```
Class(HeatSinkReynoldsNumberAnalysisModel complete
and(
  (∃ analysisrelationship.HeatSinkReynoldsNumberRelationship)
  (∀ analysisrelationship.HeatSinkReynoldsNumberRelationship)
  (∃ analysismetarelationship.ConstantFluidMaterialProperties)
  (∀ analysismetarelationship.ConstantFluidMaterialProperties)
SubClassOf(HeatSinkReynoldsNumberAnalysisModel
EquationBasedAnalysisModel)))
```

```
Class(HeatSinkReynoldsNumberRelationship partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.Velocity)
  (∃ function_of_quantity.ReynoldNumber)
  (∃ function_of_quantity.DensityAir)
  (∃ function_of_quantity.ViscosityAir)
  (∀ function_of_quantity.(DensityAir ∪ HydraulicDiameter ∪
  ReynoldNumber ∪ Velocity ∪ ViscosityAir))
  (∃ function_of_quantity.HydraulicDiameter)))
```

```
Class(ConstantFluidMaterialProperties partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.DensityAir)
  (∃ function_of_quantity.ViscosityAir)
  (∀ function_of_quantity.(DensityAir ∪ViscosityAir))
  (∃ function_of_quantity.HydraulicDiameter)))
```

6.2.1.6 Fully Developed, Convective Coefficient Analysis Model

Convective heat transfer is a function of the convective heat transfer coefficient, which depends on boundary layer, surface geometry, nature of the fluid and several other heat and mass transport properties. The Nusselt number and the convective heat transfer coefficient for non-circular ducts are obtained from Table 6-2.

$$Nu_D = \frac{hD_h}{k} \qquad \textbf{6.28}$$

Rearranging equation 6.25 results in the following relationship:

$$\bar{h} \approx \frac{Nu_D k}{D_h} \qquad \textbf{6.29}$$

Nusselt numbers for fully developed flow in channels are presented in [66]. Nusselt numbers are determined from empirically from the data given in Table 6-2.

Table 6-2: Nusselt numbers for fully developed laminar flow in tubes [66]

		$Nu_D = \frac{hD_h}{k}$	
Cross Section	$\frac{b}{a}$	**Uniform** $q_s^{"}$	**Uniform** T_s
	-	4.36	3.66
	1.0	3.61	2.98
	1.43	3.73	3.08
	2.0	4.12	3.39

	3.0	4.79	3.96
	4.0	5.33	4.44
	8.0	6.49	5.60
	∞	8.23	7.54
	-	3.11	2.47

The values presented in Table 6-2 assume a fully developed, laminar flow regime. Using the empirical data presented in Table 6-2 enables the convection coefficient to be determined knowing the relationships between the cross-sectional area of the channel. However, if Reynolds number is greater than 2300, the flow regime is no longer laminar and the empirical data presented in Table 6-2 can not be used. An analysis model is developed to determine the convection heat transfer coefficient, using the empirical data and a linear interpolation function.

$$\frac{Nu_D\big|_2 - Nu_D\big|_1}{\left.\frac{b}{a}\right|_2 - \left.\frac{b}{a}\right|_1} = \frac{Nu_D\big|_x - Nu_D\big|_1}{\left.\frac{b}{a}\right|_x - \left.\frac{b}{a}\right|_2} \qquad \textbf{6.30}$$

The analysis model for determining the heat sink convection heat transfer coefficient is illustrated graphically in Figure 6-11.

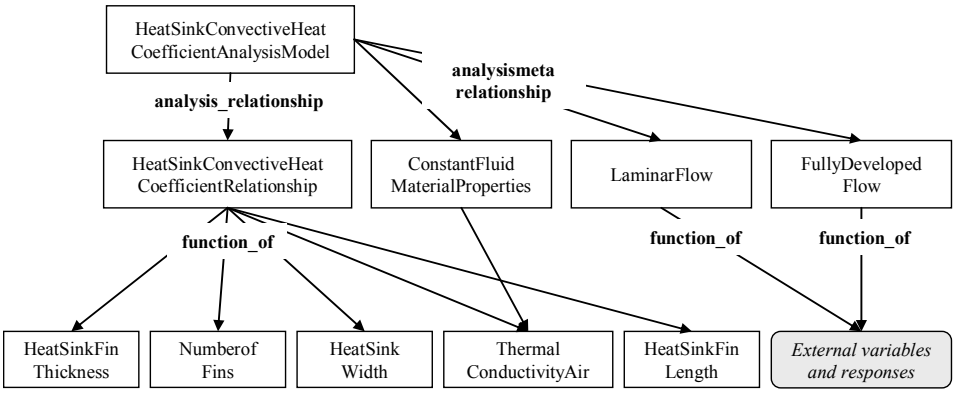


Figure 6-11: Graphical representation of convective heat transfer coefficient analysis model

As illustrated in Figure 6-11, the `HeatSinkConvectiveHeatCoefficientAnalysisModel` is comprised of a single analysis relationship and three meta-level constraints. The digital interface is defined by the following Quantity concepts: `HeatSinkFinLength`, `HeatSinkFinThickness`, `HeatSinkWidth`, `NumberOfFins`, and `ThermalConductivityAir`. In addition, the ConstantFluidMaterialProperties constraint is a function of `ThermalConductivityAir`, whereas the `LaminarFlow` and `FullyDevelopedFlow` meta-relationships are both a function of `ReynoldsNumber`. The convective heat transfer coefficient analysis constraint relationships and analysis model are implemented using DL as:

```
Class(HeatSinkConvectiveCoefficientAnalysisModel complete
and(
   (∃ analysisrelationship.HeatSinkConvectiveCoefficientRelationship)
   (∀ analysisrelationship.HeatSinkConvectiveCoefficientRelationship)
   (∃ analysismetarelationship.ConstantFluidMaterialProperties)
   (∃ analysismetarelationship.LaminarFlow)
   (∃ analysismetarelationship.FullyDevelopedFlow)
   (∀ analysismetarelationship.(ConstantFluidMaterialProperties∪
   LaminarFlow ∪ FullyDevelopedFlow)
```

```
SubClassOf(HeatSinkHydraulicDiameterAnalysisModel
   EquationBasedAnalysisModel)
```

```
Class(HeatSinkConvectiveCoefficientRelationship partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.HeatSinkFinLength)
   (∃ function_of_quantity.HeatSinkFinThickness)
   (∃ function_of_quantity.NumberOfFins)
   (∃ function_of_quantity.ThermalConductivityAir)
   (∃ function_of_quantity.HeatSinkWidth)
   (∀ function_of_quantity.(HeatSinkFinLength∪HeatSinkFinThickness∪
   HeatSinkWidth∪ ThermalConductivityAir ∪NumberOfFins))
```

```
Class(LaminarFlow partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.ReynoldNumber)
   (∀ function_of_quantity.ReynoldNumber)))
```

```
Class(FullyDevelopedFlow partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.ReynoldsNumber)
  (∀ function_of_quantity.ReynoldsNumber)))
```

6.2.2 Structural Analysis Models

In addition to considering the thermal performance of the fin array heat sink, several structural behaviors must be modeled. Fin array heat sinks are often subjected to external loading to ensure the heat sink makes adequate contact with the heat source, thus reducing thermal contact resistance and increasing total heat transfer. Mechanical clips and springs are often utilized to apply a constant force on the heat sink. A simplified model applied load is illustrated in Figure 6-12.

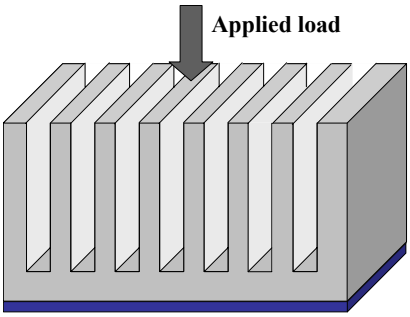


Figure 6-12: Externally applied load to heat sink

In designing fin array heat sinks deflection, stiffness, stress, and buckling phenomena must be modeled. Several structural relationships are derived for use in the engineering design decisions. The schematic for computing structural behaviors of the fin array heat sink is given in Figure 6-13.

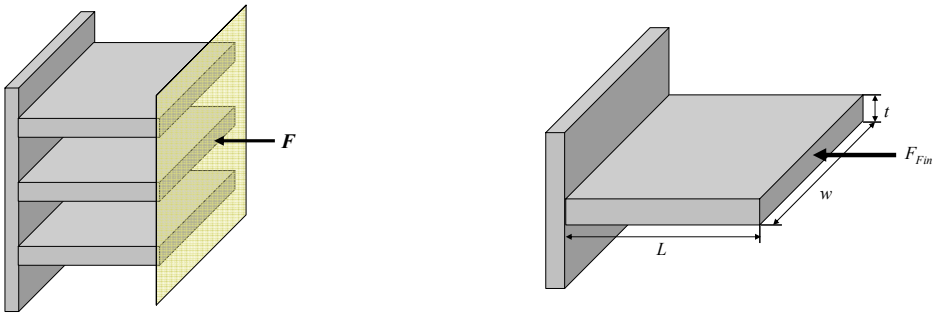


Figure 6-13: Schematic for structural analysis models

6.2.2.1 Compressive Stress Model

The load supported by each of the fins in the heat sink is computed as:

$$F_{Fin} = \frac{F}{N} \qquad \textbf{6.31}$$

where F is the total externally applied load and N is the number of fin in the array. The assumption is the load is applied uniformly over all of the fins. The stress in each of the fins is:

$$\sigma_{Fin} = \frac{F_{Fin}}{A_{Fin}} \qquad \textbf{6.32}$$

where A_{Fin} is the area of a single fin given as:

$$A_{Fin} = t \cdot L \qquad \textbf{6.33}$$

The relationship for computing stress in fin assumes there is no plastic deformation, the beam is homogeneous, the deflections are small, and buckling does not occur. Thus, for the stress relationship to be valid, a number of other analysis models must be checked. The graphical representation of the compressive stress in the fin is illustrated in Figure 6-14.

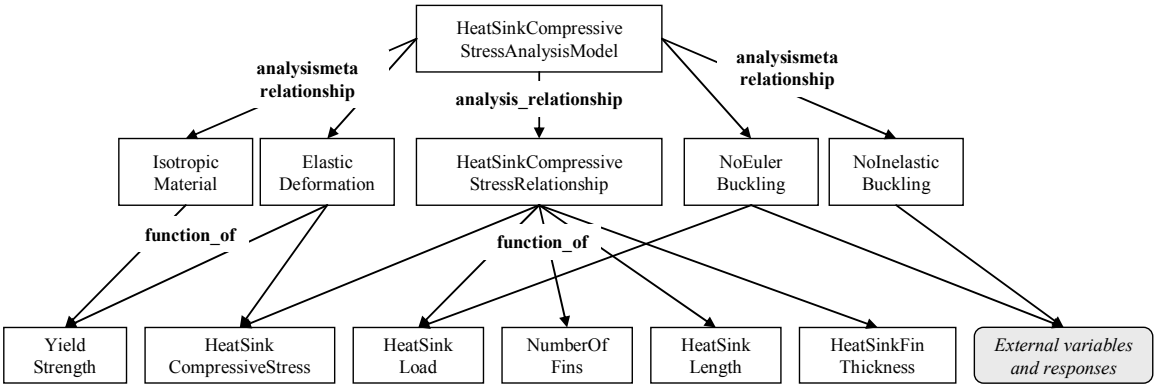


Figure 6-14: Graphical representation of compressive stress analysis model

The `HeatSinkCompressiveStressAnalysisModel` is defined with a single analysis relationship, `HeatSinkCompressiveStressAnalysisRelationship` and four meta-level relationships including: `IsotropicMaterial`, `ElasticDeformation`, `NoEulerBuckling`, and `NoInelasticBuckling`. The compressive stress in the fins of the heat sink is computed as a function of the `HeatSinkLoad`, `NumberOfFins`, `HeatSinkLength`, and

HeatSinkFinThickness. Additionally, the meta-level constraints require additional information to be instantiated for the model including: YieldStrength, EulerCriticalBucklingLoad, and HeatSinkInelasticBuckling. Thus, the digital interface to the analysis model is the aggregation of the quantities and is defined as: YieldStrength, HeatSinkCompressiveStress, HeatSinkLoad, NumberOfFins, HeatSinkLength, HeatSinkFinThickness, EulerCriticalBucklingLoad, and HeatSinkInelasticBuckling. The compressive stress analysis model is defined in DL as:

```
Class(HeatSinkCompressiveStressAnalysisModel complete(
and(
   (∃ analysisrelationship.HeatSinkCompressiveStressRelationship)
   (∃ analysismetarelationship.IsotropicMaterialProperties)
   (∃ analysismetarelationship.NoElasticBuckling)
   (∃ analysismetarelationship.NoEulerBuckling)
   (∃ analysismetarelationship.ElasticDeformation)
   (∀ analysismetarelationship.(ElasticDeformation ∪
   IsotropicMaterialProperties ∪ NoElasticBuckling ∪ NoEulerBuckling))

SubClassOf(HeatSinkCompressiveStressAnalysisModel
   EquationBasedAnalysisModel)))
```

```
Class(HeatSinkCompressiveStressRelationship partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.HeatSinkLoad)
   (∃ function_of_quantity.NumberOfFins)
   (∃ function_of_quantity.HeatSinkLength)
   (∃ function_of_quantity.HeatSinkFinThickness)
   (∃ function_of_quantity.HeatSinkCompressiveStress)
   (∀ function_of_quantity.(HeatSinkFinThickness ∪ HeatSinkLength ∪
   HeatSinkLoad ∪ NumberOfFins ∪ HeatSinkCompressiveStress))))
```

```
Class(IsotropicMaterialProperties partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.HeatSinkYieldStrength)
  (∀function_of_quantity.HeatSinkYieldStrength)))
```

```
Class(ElasticDeformation partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity. HeatSinkYieldStrength)
  (∃ function_of_quantity. HeatSinkCompressiveStress)
  (∀function_of_quantity.( HeatSinkYieldStrength ∪
  HeatSinkCompressiveStress))))
```

```
Class(NoEulerBuckling partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.EulerCriticalLoad)
  (∃ function_of_quantity. BeamNormalStress)
  (∀ function_of_quantity.(EulerCriticalLoad ∪ BeamNormalStress))))
```

```
Class(NoElasticBuckling partial
and(
  EquationBasedConstraintRelationship
  (∃ function_of_quantity.InelasticCriticalLoad)
  (∃ function_of_quantity. BeamNormalStress)
  (∀ function_of_quantity.(InelasticCriticalLoad ∪ BeamNormalStress))))
```

6.2.2.2 Vertical Stiffness Analysis Model

The stiffness equation for a compression member is given as:

$$k_{fin} = \frac{A_{fin} E}{l_{fin}} \qquad \textbf{6.34}$$

$$K = \sum_{i=1}^{N} k_i \qquad \textbf{6.35}$$

Similar to the stress relationship, the stiffness relationship is valid for several assumptions including: the stresses are within the elastic limit, the beam is homogeneous, the deflections are small, and buckling does not occur. The information associated with the vertical stiffness analysis model is illustrated graphically in Figure 6-15.

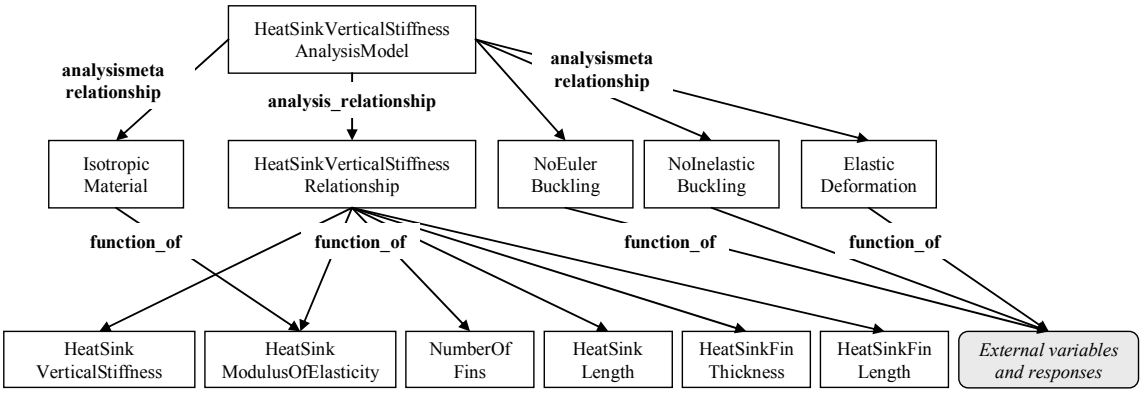


Figure 6-15: Graphical representation of heat sink vertical stiffness analysis model

The `HeatSinkVerticalStiffnessAnalysisModel` is similar to the `HeatSinkCompressiveStressAnalysisModel` concept. The stiffness of the heat sink is computed through the `HeatSinkVerticalStiffnessRelationship`. The analysis model is constrained by the same meta-level constraints associated with the `HeatSinkCompressiveStressAnalysisModel` including `IsotropicMaterial`, `ElasticDeformation`, `NoEulerBuckling`, and `NoInelasticBuckling`. Thus, the resulting digital interface for computing the stiffness of the heat sink is defined as: `YieldStrength`, `HeatSinkVerticalStiffness`, `HeatSinkLoad`, `NumberOfFins`, `HeatSinkLength`, `HeatSinkFinThickness`, `EulerCriticalBucklingLoad`, and `HeatSinkInelasticBuckling`. A minor difference in the two models is the instantiation of the elastic stress constraint. In the stiffness analysis model the elastic stress must be computed and checked in an external analysis model. In the compressive stress analysis model, the elastic stress meta-constraint is determined within the scope of the analysis model. A DL representation of

the vertical stiffness analysis model is developed using the graphical representation presented in Figure 6-15.

```
Class(HeatSinkVerticalStiffnessAnalysisModel complete(
and(
   (∃ analysisrelationship.HeatSinkVerticalStiffnessRelationship)
   (∀ analysisrelationship.HeatSinkVerticalStiffnessRelationship)
   (∃ analysismetarelationship.IsotropicMaterialProperties)
   (∃ analysismetarelationship.NoElasticBuckling)
   (∃ analysismetarelationship.NoEulerBuckling)
   (∃ analysismetarelationship.ElasticDeformation)
   (∀ analysismetarelationship.(ElasticDeformation ∪
   IsotropicMaterialProperties ∪ NoElasticBuckling ∪ NoEulerBuckling))

SubClassOf(HeatSinkVerticalStiffnessAnalysisModel
   EquationBasedAnalysisModel))
```

```
Class(HeatSinkVerticalStiffnessRelationship partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.HeatSinkModulusOfElasticity)
   (∃ function_of_quantity.NumberOfFins)
   (∃ function_of_quantity.HeatSinkLength)
   (∃ function_of_quantity.HeatSinkVerticalStiffness)
   (∃ function_of_quantity.HeatSinkFinThickness)
   (∃ function_of_quantity.HeatSinkFinLength)
   (∀ function_of_quantity.(HeatSinkFinLength ∪ HeatSinkFinThickness ∪
   HeatSinkLength ∪ HeatSinkModulusOfElasticity ∪
   HeatSinkVerticalStiffness ∪ NumberOfFins)))))
```

6.2.2.3 Euler Buckling Model

Buckling of the fin in the heat sink is modeled using the Euler Buckling formula given as:

$$P_{critical} = \frac{\pi^2 EI}{L_{eff}^{\ 2}} \qquad \textbf{6.36}$$

where L_{eff} is the effective length. The effective length of the fin in the heat sink is modeled as a fixed-free column and is computed as:

$$L_{eff} = 2l_{fin} \qquad \textbf{6.37}$$

For a rectangular fin, I is computed as:

$$I = \frac{1}{12} L t^3 \qquad \textbf{6.38}$$

The Euler buckling formula is valid for *long columns* only. However, a different relationship must be used for short and intermediate columns. The Euler buckling model of the fin is illustrated graphically in Figure 6-16

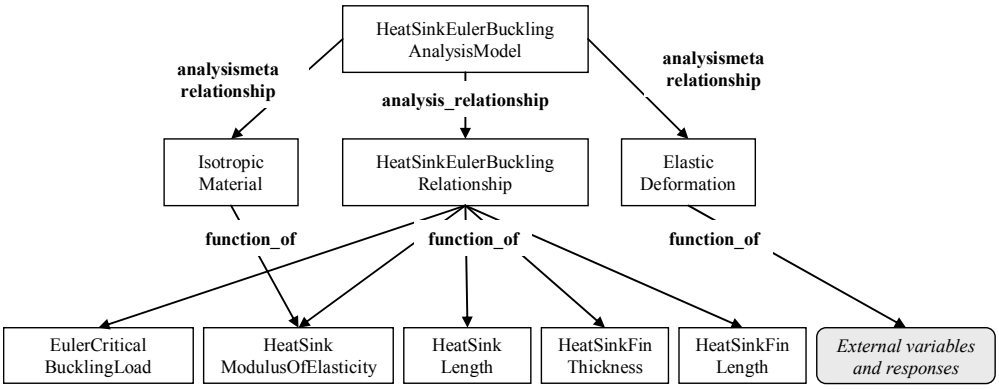


Figure 6-16: Graphical representation of heat sink Euler buckling analysis model

The `HeatSinkEulerBucklingAnalysisModel` is comprised of a single `analysis_relationship` and two `analysismetarelationships`. The Euler buckling analysis model is constrained by two assumptions: `IsotropicMaterial` and `ElasticDeformation`. These meta-level relationships are reused from the previously defined concept definitions (see Section 6.2.2.1). The critical Euler buckling load for the heat sink is computed using the `HeatSinkEulerBucklingRelationship`. The required information for using

the analysis model includes: HeatSinkModulusOfElasticity, YieldStrength, HeatSinkLoad, HeatSinkLength, HeatSinkFinThickness, and HeatSinkCompressiveStress. The Euler buckling analysis model is defined using DL as:

```
Class(HeatSinkEulerBucklingAnalysisModel complete
and(
   (∃ analysisrelationship.HeatSinkEulerBucklingRelationship)
   (∀ analysisrelationship.HeatSinkEulerBucklingRelationship)
   (∃ analysismetarelationship.IsotropicMaterialProperties)
   (∃ analysismetarelationship.ElasticDeformation)
   (∀ analysismetarelationship.(ElasticDeformation ∪
   IsotropicMaterialProperties))

SubClassOf(HeatSinkEulerBucklingAnalysisModel
   EquationBasedAnalysisModel)
```

```
Class(HeatSinkEulerBucklingRelationship partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.HeatSinkModulusOfElasticity)
   (∃ function_of_quantity.HeatSinkLength)
   (∃ function_of_quantity.HeatSinkFinThickness)
   (∃ function_of_quantity.HeatSinkFinLength)
   (∃ function_of_quantity.EulerCriticalBucklingLoad)
   (∀ function_of_quantity.(HeatSinkFinLength ∪ HeatSinkFinThickness ∪
   HeatSinkLength ∪ HeatSinkModulusOfElasticity ∪
   EulerCriticalBucklingLoad))))
```

6.2.2.4 Inelastic Buckling Model

Inelastic buckling is occurs when the stress in the column exceeds the elastic limit of the material in the column before the critical (Euler) stress and the ultimate strength of the material is reached (see Figure 6-17)

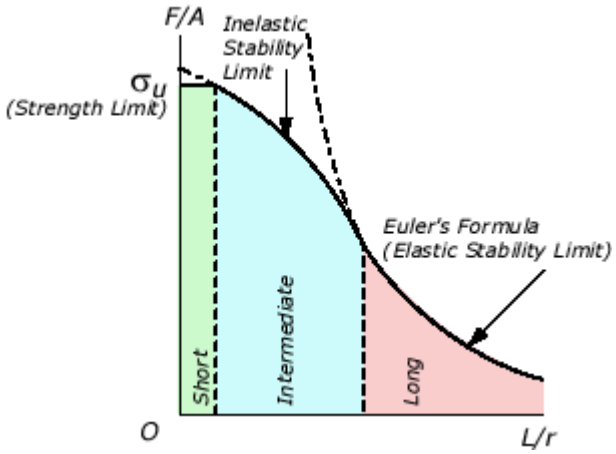


Figure 6-17: Empirical relationship for determining column buckling [42]

The buckling load for intermediate is related to the geometry of the column and the material strength. The typical limits for column buckling are presented in Table 6-3.

Table 6-3: Ranges for buckling of short, intermediate and long columns [42]

Material	Short Column (Strength Limit)	Intermediate Column (Inelastic Stability Limit)	Long Column (Elastic Stability Limit)
	Slenderness Ratio ($SR = L_{eff}/r$)		
Structural Steel	$SR < 40$	$40 < SR < 150$	$SR > 150$
Aluminum Alloy AA 6061 - T6	$SR < 9.5$	$9.5 < SR < 66$	$SR > 66$
Aluminum Alloy AA 2014 - T6	$SR < 12$	$12 < SR < 55$	$SR > 55$
Wood	$SR < 11$	$11 < SR < (18 \sim 30)$	$(18 \sim 30) < SR < 50$

The slenderness ratio of the column is computed as:

$$SR = \frac{L_{eff}}{r_g} \qquad \textbf{6.39}$$

where r_g is the radius of gyration and given as:

$$r_g = \sqrt{\frac{I}{A}} \qquad \textbf{6.40}$$

A similar approach to that used in Section 6.2.1.6 is employed to determine the inelastic buckling of the fin. Empirical data is interpolated to determine the critical conditions for inelastic buckling to occur. The information associated with the inelastic buckling model is represented graphically in Figure 6-18.

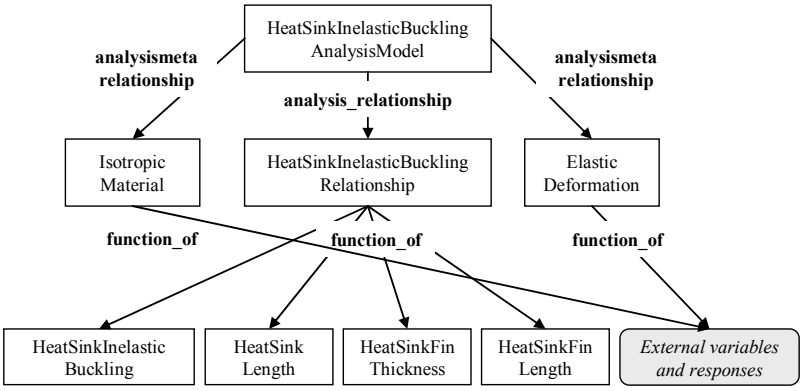


Figure 6-18: Graphical representation of heat sink inelastic buckling analysis model

The `HeatSinkInelasticBucklingAnalysisModel` is comprised of a single `analysis_relationship` and two `analysismetarelationships`. The inelastic buckling model is constrained by the same meta-level relationships as the Euler buckling analysis model. The `HeatSinkInelasticBucklingAnalysisModel` digital interface is defined by variables directly associated with the analysis models including `HeatSinkInelasticBuckling`, `HeatSinkLength`, `HeatSinkFinThickness`, `HeatSinkFinLength`, `ModulusOfElasticity`, `YieldStrength`, and `HeatSinkLoad` and variables computed in external analysis models including heat sink material properties,

YieldStrength, and the HeatSinkCompressiveStress. The inelastic buckling analysis model is defined using DL as:

```
Class(HeatSinkInelasticBucklingAnalysisModel complete
and(
   (∃ analysisrelationship.HeatSinkEulerBucklingRelationship)
   (∀ analysisrelationship.HeatSinkEulerBucklingRelationship)
   (∃ analysismetarelationship.IsotropicMaterialProperties)
   (∃ analysismetarelationship.ElasticDeformation)
   (∀ analysismetarelationship.(ElasticDeformation ∪
   IsotropicMaterialProperties))

SubClassOf(HeatSinkEulerBucklingAnalysisModel
   EquationBasedAnalysisModel)
```

```
Class(HeatSinkInelasticBucklingRelationship partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.HeatSinkLength)
   (∃ function_of_quantity.HeatSinkInelasticBuckling)
   (∃ function_of_quantity.HeatSinkFinThickness)
   (∃ function_of_quantity.HeatSinkFinLength)
   (∀ function_of_quantity.(HeatSinkFinLength ∪ HeatSinkFinThickness ∪
   HeatSinkLength ∪ HeatSinkInelasticBuckling))))
```

6.2.3 General Fin Array Heat Sink Models

For lack of a better categorization, the analysis models presented in this section are used to determine form-based characteristics of the heat sink. As shown in this section, the mass and the volume of the heat sink are directly related to the shape and material of the heat sink.

6.2.3.1 Fin array Heat Sink Mass Analysis Model

The mass of the heat sink is given as:

$$M_{Heatsink} = \rho_{Heatsink}\left(\left(N \cdot L \cdot t \cdot l_{fin}\right) + \left(L \cdot W \cdot t_{base}\right)\right) \qquad \textbf{6.41}$$

where $\rho_{Heatsink}$ is the density of the heat sink material. The mass equation assumes a constant density. The information associated with the heat sink mass model is represented graphically in Figure 6-19.

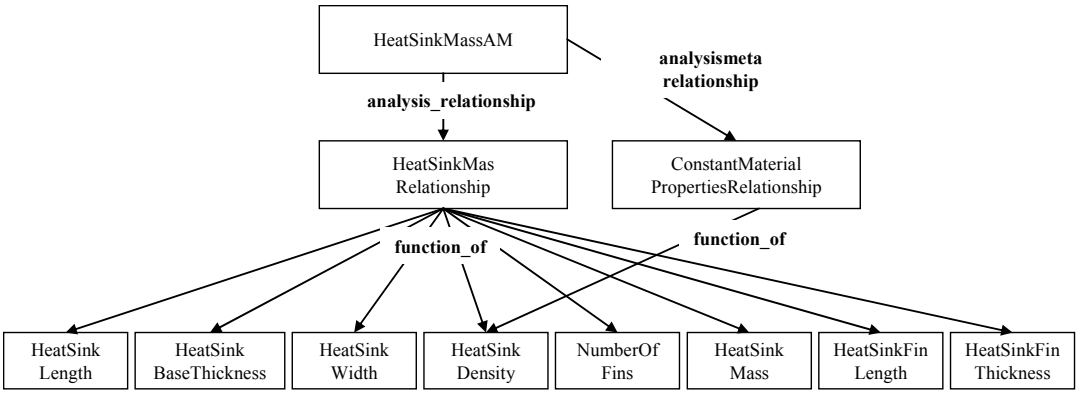


Figure 6-19: Graphical representation of heat sink mass analysis model

The heat sink mass analysis model is characterized by a single analysisrelationship and one meta-level relationship that limit the applicability of the analysis models. For example, the `ConstantMaterialPropertiesRelationhip` requires that the heat sink be constructed of a material with a constant set of properties. The density of the material is the primary concern in computing the mass of the heat sink. For example, the constant material property assumption is a blanket constraint applied to the analysis model for all material properties. The heat sink mass analysis model is implemented using the DL-based formal language as:

```
Class(HeatSinkMassAnalysisModel complete
and(
   (∃ analysisrelationship.HeatSinkMassRelationship)
   (∀ analysisrelationship.HeatSinkMassRelationship)
   (∃ analysismetarelationship.ConstantMaterialPropertiesRelationship)
   (∀ analysismetarelationship.ConstantMaterialPropertiesRelationship)

SubClassOf(HeatSinkMassAnalysisModel EquationBasedAnalysisModel)))
```

```
Class(HeatSinkMassRelationship partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.HeatSinkLength)
   (∃ function_of_quantity.HeatSinkFinThickness)
   (∃ function_of_quantity.HeatSinkFinLength)
   (∃ function_of_quantity.HeatSinkWidth)
   (∃ function_of_quantity.NumberOfFins)
   (∃ function_of_quantity.HeatSinkDensity)
   (∃ function_of_quantity.HeatSinkBaseThickness)
   (∃ function_of_quantity.HeatSinkMass)

   (∀ function_of_quantity.(HeatSinkFinLength ∪ HeatSinkFinThickness ∪
   HeatSinkLength ∪ HeatSinkWidth∪NumberOfFins∪HeatSinkDensity∪
   HeatSinkBaseThickness∪ HeatSinkMass))))
```

6.2.3.2 Fin array Heat Sink Space Analysis Model

The volume that the heat sink determined using the following equation:

$$Vol_{Heatsink} = W \cdot L \cdot \left(t_{base} + l_{fin} \right) \qquad \textbf{6.42}$$

Using Equation 6.42, the formal language is used to explicitly represent the information structure associated with the heat sink volume analysis model. The graphical representation of the heat sink volume analysis model is presented in Figure 6-20.

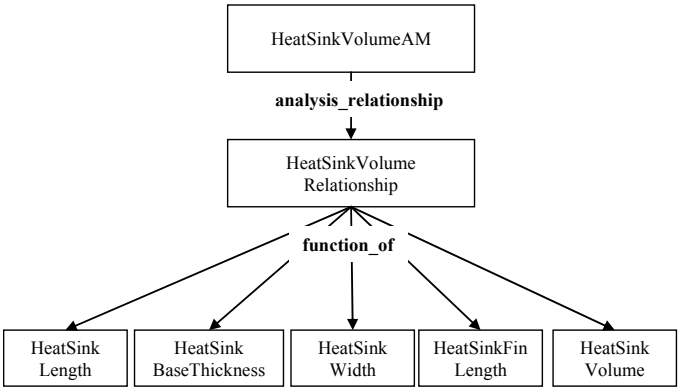


Figure 6-20: Graphical representation of heat sink volume analysis model

As illustrated in Figure 6-20, the volume analysis model implemented in this dissertation does not have any meta-level assumptions specified. Additionally, the HeatSinkVolume is computed as a function of HeatSinkLength, HeatSinkBaseThickness, HeatSinkWidth, and HeatSinkFinLength. Thus the digital interface through which information is passed is defined as: HeatSinkVolume, HeatSinkLength, HeatSinkBaseThickness, HeatSinkWidth, and HeatSinkFinLength. The DL representation of the heat sink volume analysis models is given as:

```
Class(HeatSinkVolumeAnalysisModel complete
and(
   (∃ analysisrelationship.HeatSinkVolumeRelationship)
   (∀ analysisrelationship.HeatSinkVolumeRelationship)

SubClassOf(HeatSinkVolumeAnalysisModel EquationBasedAnalysisModel)))
```

The HeatSinkVolumeAnalysisModel is composed of a single EquationBasedConstraintRelationship concept given as:

```
Class(HeatSinkVolumeMassRelationship partial
and(
   EquationBasedConstraintRelationship
   (∃ function_of_quantity.HeatSinkLength)
   (∃ function_of_quantity.HeatSinkFinLength)
   (∃ function_of_quantity.HeatSinkWidth)
   (∃ function_of_quantity.HeatSinkBaseThickness)
   (∃ function_of_quantity.HeatSinkVolume)

   (∀ function_of_quantity.(HeatSinkFinLength ∪ HeatSinkLength ∪
   HeatSinkWidth ∪ HeatSinkBaseThickness∪ HeatSinkVolume))))
```

In the following section, the analysis models are integrated into cDSP concepts and relationships are established to facilitate the integration and exchange of information across disciplinary analysis models and multi-disciplinary decision information.

6.3 Information Modeling of Heat Sink Design Decision Problems

In this section, the representation of decision-related information and the subsequent integration of disciplinary analysis models are presented. Three different heat sink design problems are discussed. The first is a thermal design problem, the second is a structural design problem, and the third represents the integration of structural and thermal design considerations. The formal language is used in conjunction with the previously defined analysis models to capture the information associated with the design problem.

6.3.1 Heat Sink Design Problem Example 1 – Thermal Design Decision

The first example of the heat sink design problem takes into account only the thermal aspects of the heat sink. The design goals are to minimize volume and thermal resistance. The heat sink design problem is described in Table 6-4. The design problem is represented as a cDSP in Figure 6-21.

Table 6-4: Fin array heat sink design problem description – Example 1

- The system variables include: number of fins, fin length, and fin thickness
- The design parameters include: ambient air temperature, volumetric flow rate of the cooling air, material properties of the heat sink, material properties of the cooling air, the power dissipated from the CPU, the width of the heat sink, the depth of the heat sink
- The design objective is to minimize the thermal resistance, volume, and mass of the heat sink
- The fin array heat sink must remain within specified volumetric keep-in space based on the footprint of the CPU 80 mm × 80 mm × 50 mm
- The Mass of the heat sink should minimized to ensure that mechanical shock and vibration are minimized to the CPU, the target Mass is 0.01 kg
- The maximum steady-state temperature in the heat sink must not exceed 70 degrees C (343 K).

GIVEN

Design parameters

- Heat sink geometry parameters
- Heat sink material properties
- Cooling fluid material properties
- Boundary conditions

Disciplinary Analysis Models

- Heat transfer analysis models
  1. Thermal resistance
  2. Maximum temperature
  3. Convection heat transfer coefficient
- Fluid analysis models
  4. Reynolds number
  5. Hydraulic diameter
  6. Fluid velocity
- Packaging analysis models
  7. Heat sink Mass
  8. Heat sink volume

FIND

System variables

- Number of fins
- Fin length
- Fin thickness

Deviation variables

- Deviation from target Mass
- Deviation from target thermal resistance

SATISFY

System bounds

- Bounds on fin width, $0.0001m \le t_{fin} \le 0.0005m$
- Bounds on fin length, $0.01m \le l_{fin} \le 0.05m$
- Bounds on number of fins, $2 \le N \le 30$

System design requirements & constraints

- Chip temperature is below maximum temperature

Analysis models constraints & limitations

- *Model limitations imposed by analysis models*

System Goals

- Minimize deviation of heat sink mass from target mass,
  $M_{Heatsink} = f\left(N, L, t, l_{fin}, \rho_{Heatsink}, W, t_{base}\right)$
- Minimize deviation of heat sink thermal resistance from target thermal resistance,
  $R_t = f\left(W, t, k, t_{base}, \overline{h}, l_{fin}, N, L\right)$
- Minimize deviation of heat sink volume from target volume, $Vol_{Heatsink} = f\left(W, L, t_{base}, l_{fin}\right)$

Decision constraints

- $d_i^+, d_i^- \ge 0$ & $d_i^+ \cdot d_i^- = 0$ for all design objectives

MINIMIZE

Deviation function - $Z = f\left(w_{mass} \cdot d_{mass}^- + w_{resistance} \cdot d_{resistance}^- + w_{volume} \cdot d_{volume}^-\right)$

Figure 6-21: Fin array heat sink cDSP - Example 1

A graphical illustration of the cantilever beam design problem 1 is presented in Figure 6-22.

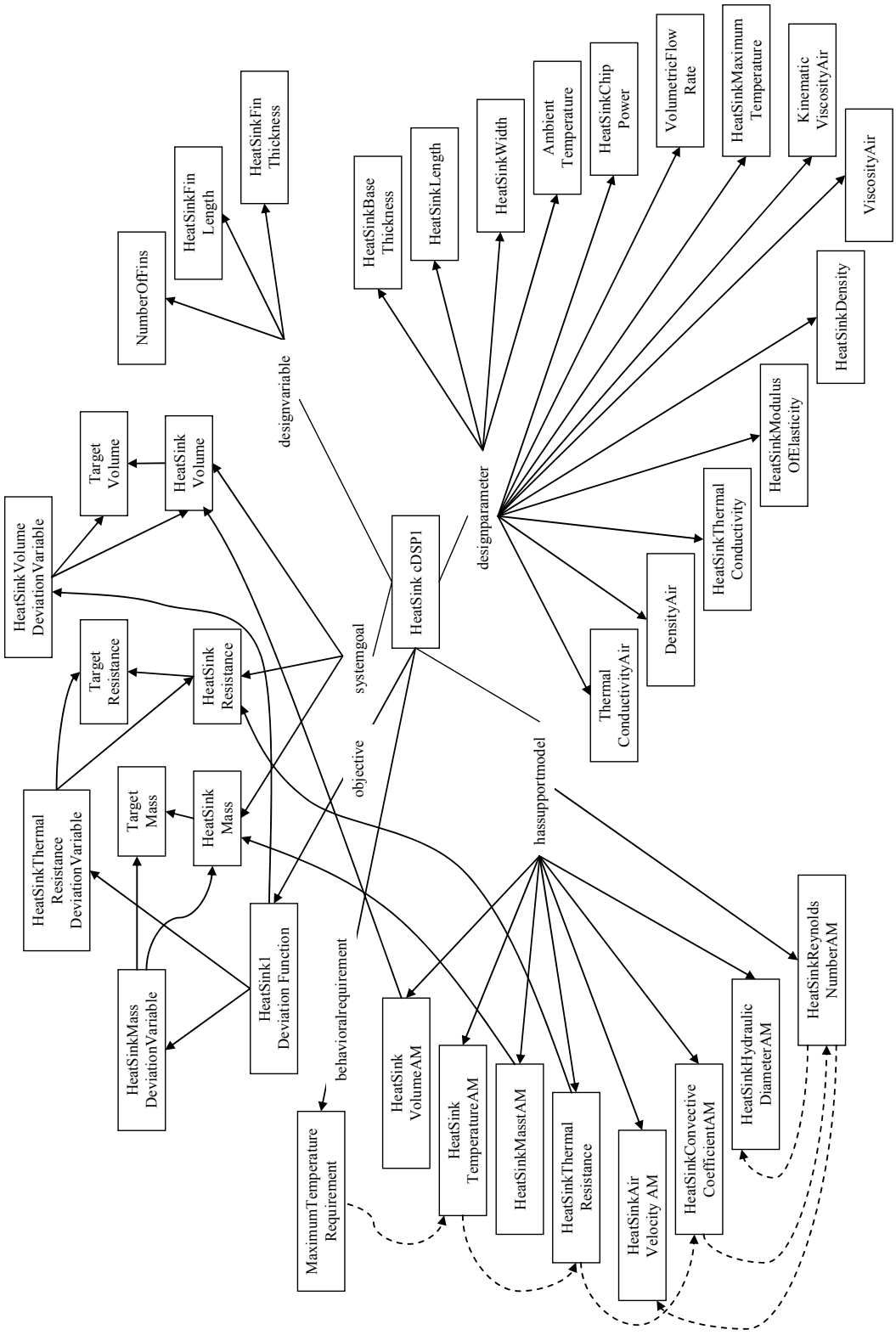


Figure 6-22: Graphical representation of heat sink design problem 1

As illustrated in Figure 6-22 the cDSP for Example Design Problem 1 consists of fourteen designparameters, three design variables, three design goals, eight analysis models, one behavioralrequirement, and a single deviation function. While all of the relationships are not represented in Figure 6-22, sufficient detail is represented to gain an understanding of the dependencies and coupling of analysis models and decision quantities. For example, the relationships (i.e., the chained analysis models) are represented in Figure 6-22 using dashed lines. Additionally, the relationships between the analysis models and the designgoals and behavioralrequirements are captured. For example, the MaximumTemperatureRequirement is verified through the HeatSinkTemperatureAM and the HeatSinkMaximumTemperature quantity. The implied relationships between the analysis models and the designvariables and designparameters are not captured explicitly in Figure 6-22. A DL representation of the heat sink cDSP is developed based on the graphical representation and defined as:

```
Class (HeatSinkCDSP1
and(
   (∃ designvariable.NumberOfFins)
   (∃ designvariable.HeatSinkFinLength)
   (∃ designvariable.HeatSinkFinThickness)
   (∃ designparameter.ThermalConductivityAir)
   (∃ designparameter.DensityAir)
   (∃ designparameter.HeatSinkThermalConductivity)
   (∃ designparameter.HeatSinkModulusOfElasticity)
   (∃ designparameter.HeatSinkDensity)
   (∃ designparameter.ViscosityAir)
   (∃ designparameter.KinematicViscosityAir)
   (∃ designparameter.HeatSinkMaximumTemperature)
   (∃ designparameter.VolumetricFlowRate)
   (∃ designparameter.HeatSinkChipPpower)
   (∃ designparameter.AmbientTemperature)
   (∃ designparameter.HeatSinkWidth)
   (∃ designparameter.HeatSinkLength)
   (∃ designparameter.HeatSinkBaseThickness)
```

```
(∃ hassupportmodel.HeatSinkVolumeAM)
(∃ hassupportmodel.HeatSinkTemperatureAM)
(∃ hassupportmodel.HeatSinkThermalResistanceAM)
(∃ hassupportmodel.HeatSinkAirVelocityAM)
(∃ hassupportmodel.HeatSinkConvectiveCoefficientAM)
(∃ hassupportmodel.HeatSinkHydraulicDiameterAM)
(∃ hassupportmodel.HeatSinkReynoldsNumberAM)
(∃ hassupportmodel.HeatSinkMassAM)

(∃ systemgoal.(HeatSinkVolume ∩
  (= targetsystembehavior.1) ∩
  (∃ targetsystembehavior.TargetVolume) ∩
  (∀ targetsystembehavior.TargetVolume)))

(∃ systemgoal.(HeatSinkMass ∩
  (= targetsystembehavior 1) ∩
  (∃ targetsystembehavior.TargetMass) ∩
  (∀ targetsystembehavior.TargetMass)))
(∃ systemgoal.(HeatSinkThermalResistance ∩
  (= targetsystembehavior 1) ∩
  (∃ targetsystembehavior.TargetThermalResistance) ∩
  (∀ targetsystembehavior.TargetThermalResistance)))
(= systemgoal 2))

(∃ objective.(HeatSink1DeviationFunction ∩
 (∃ function_of.( HeatSinkVolumeDeviationVariable ∩
    ((∃ relativeimportance.RelativeImportance) ∩
    (∀ relativeimportance.RelativeImportance)∩
    (= relativeimportance 1)) ∩
    ((∃ function_of.HeatSinkVolume) ∩
    (∃ function_of.TargetVolume) ∩
    (∀ function_of.(HeatSinkVolume ∪TargetVolume)))

 (∃ function_of.(HeatSinkMassDeviationVariable ∩
    ((∃ relativeimportance.RelativeImportance) ∩
    (∀ relativeimportance.RelativeImportance)∩
    (= relativeimportance 1)) ∩
    ((∃ function_of.HeatSinkMass) ∩
    (∃ function_of.TargetMass) ∩
    (∀ function_of.(HeatSinkMass∪TargetMass)))
 (∃ function_of.(HeatSinkThermalResistanceDeviationVariable ∩
    ((∃ relativeimportance.RelativeImportance) ∩
    (∀ relativeimportance.RelativeImportance)∩
    (= relativeimportance 1)) ∩
    ((∃ function_of.HeatSinkThermalResistanc) ∩
    (∃ function_of.TargetThermalResistance) ∩
    (∀ function_of.(HeatSinkThermalResistance∪
          TargetThermalResistance))))
```

```
(∀ objective.(HeatSink1DeviationFunction ∩
  (∃ function_of.( HeatSinkVolumeDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.HeatSinkVolume) ∩
     (∃ function_of.TargetVolume) ∩
     (∀ function_of.(HeatSinkVolume ∪TargetVolume)))
  (∃ function_of.(HeatSinkMassDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.HeatSinkMass) ∩
     (∃ function_of.TargetMass) ∩
     (∀ function_of.(HeatSinkMass∪TargetMass)))
  (∃ function_of.(HeatSinkThermalResistanceDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.HeatSinkThermalResistanc) ∩
     (∃ function_of.TargetThermalResistance) ∩
     (∀ function_of.(HeatSinkThermalResistance∪
     TargetThermalResistance))))
  (= objective 1)
(∃ behavioralrequirement.maximum_temperature_requirement)))
```

6.3.2 Heat Sink Design Problem Example 2 – Structural Design Decision

The second example of the heat sink design problem takes into account only the structural aspects of the heat sink. The design goals are to minimize volume and mass. The heat sink design problem is described in Table 6-5 and as a cDSP in Figure 6-23.

Table 6-5: Fin array heat sink design problem description – Example 2

- The system variables include: number of fins, fin length, and fin thickness
- The design parameters include: material properties of the heat sink, the width of the heat sink, the depth of the heat sink, heat sink load
- The design objectives are to minimize the mass and volume of the heat sink
- The fin array heat sink must remain within specified volumetric keep-in space based on the footprint of the CPU 80 mm × 80 mm × 50 mm
- The vertical stiffness of the heat sink is specified to be greater than 5,250,000 N/m to ensure proper preload and adequate contact
- The mass of the heat sink should minimized to ensure that mechanical shock and vibration are minimized to the CPU, the target mass is 0.01 kg
- The heat sink must be able to withstand a total clamping load of 300 N

GIVEN

Design parameters

- Heat sink geometry parameters
- Heat sink material properties
- Loading & boundary conditions

Disciplinary Analysis Models

- Structural analysis models
  1. Vertical stiffness
  2. Compressive Stress
  3. Euler Buckling
  4. Inelastic Buckling
- Packaging analysis models
  5. Heat sink mass
  6. Heat sink volume

FIND

System variables

- Number of fins
- Fin length
- Fin thickness

Deviation variables

- Deviation from target mass
- Deviation from target volume

SATISFY

System bounds

- Bounds on fin width, $0.0001m \le t_{fin} \le 0.0005m$
- Bounds on fin length, $0.01m \le l_{fin} \le 0.05m$
- Bounds on number of fins, $2 \le N \le 30$

System design requirements & constraints

- Heat sink stiffness greater than minimum stiffness
- Must be able to support heat sink load

Analysis models constraints & limitations

- *Model limitations imposed by analysis models*

System Goals

- Minimize deviation of heat sink mass from target mass, $M_{Heatsink} = f\left(N, L, t, l_{fin}, \rho_{Heatsink}, W, t_{base}\right)$
- Minimize deviation of heat sink volume from target volume, $Vol_{Heatsink} = f\left(W, L, t_{base}, l_{fin}\right)$

Decision constraints

- $d_i^+, d_i^- \ge 0$ & $d_i^+ \cdot d_i^- = 0$ for all design objectives

MINIMIZE

Deviation function - $Z = f\left(w_{mass} \cdot d_{mass}^- + w_{volume} \cdot d_{volume}^-\right)$

Figure 6-23: Fin array heat sink cDSP - Example 2

A graphical illustration of the cantilever beam design problem 2 is presented in Figure 6-24.

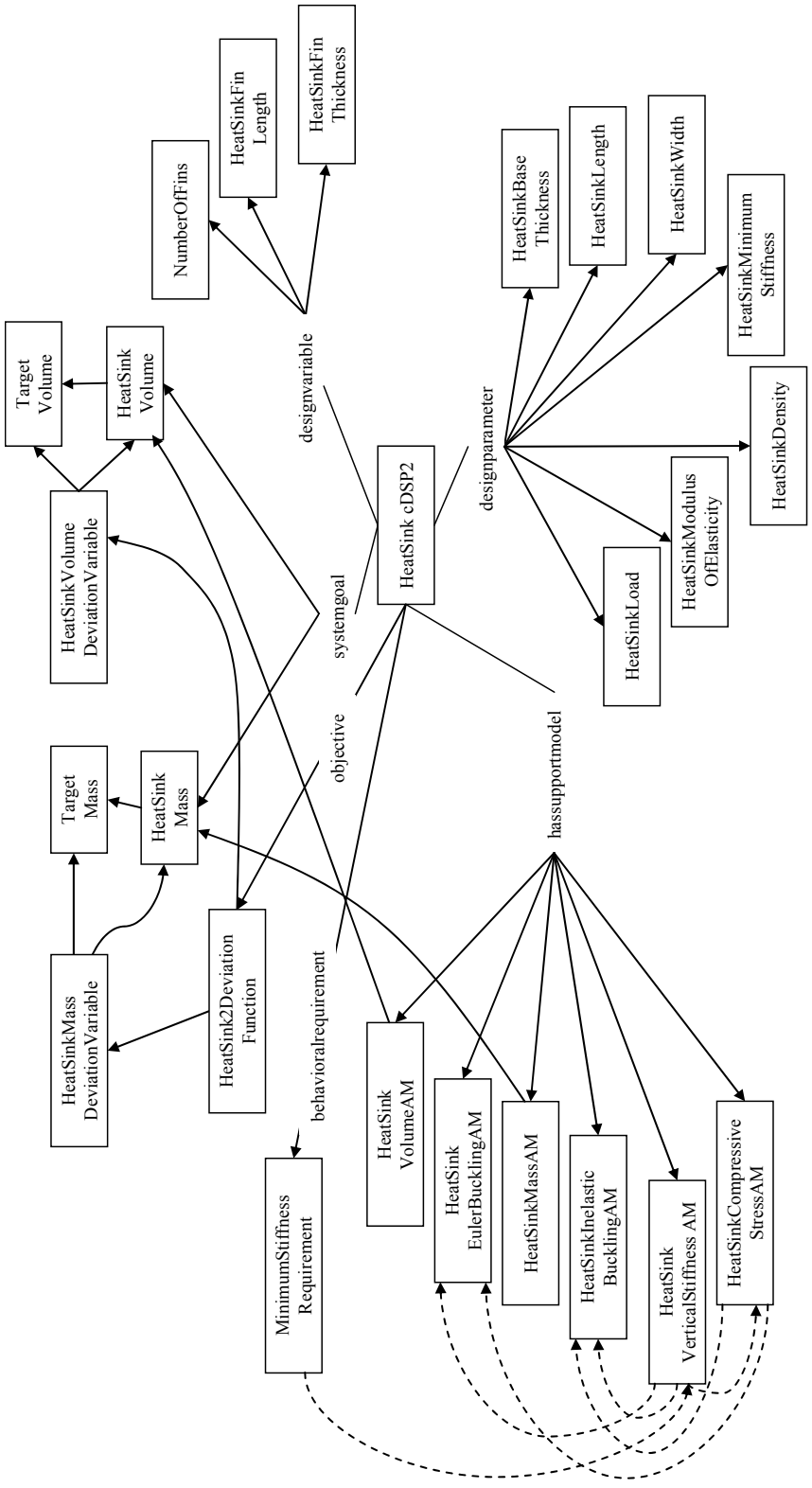


Figure 6-24: Graphical representation of heat sink design problem 2

As illustrated in Figure 6-24 the cDSP for Example Design Problem 2 consists of seven designparameters, three design variables, two design goals, six structural analysis models, one behavioralrequirement, and a single deviation function. The couplings between the analysis models are illustrated as dashed lines and represent an exchange of information and a dependency. Similarly, the systemgoals are related to specific analysis models. The ManimumStiffnessRequirement is related to the HeatSinkVerticalStiffnessAM and the HeatSinkMinimumStiffness quantity. Several implicit relationships between the analysis models and the designvariables and designparameters are not captured in Figure 6-24 for brevity and clarity. A DL implementation of the heat sink problem 2 is specified as:

```
Class (HeatSinkCDSP2
and(
   (∃ designvariable NumberOfFins)
   (∃ designvariable HeatSinkFinLength)
   (∃ designvariable HeatSinkFinThickness)
   (∃ designparameter HeatSinkDensity)
   (∃ designparameter HeatSinkMinimumStiffness)
   (∃ designparameter HeatSinkModulusOfElasticity)
   (∃ designparameter HeatSinkLoad)
   (∃ designparameter HeatSinkWidth)
   (∃ designparameter HeatSinkLength)
   (∃ designparameter HeatSinkBaseThickness)
   (∃ hassupportmodel HeatSinkCompressiveStressAM)
   (∃ hassupportmodel HeatSinkVerticalStiffnessAM)
   (∃ hassupportmodel HeatSinkInelasticBucklingAM)
   (∃ hassupportmodel HeatSinkEulerBucklingAM)
   (∃ hassupportmodel HeatSinkVolumeAM)
   (∃ hassupportmodel HeatSinkMassAM)
   (∃ systemgoal (HeatSinkMass ∩
     (= targetsystembehavior 1) ∩
     (∃ targetsystembehavior TargetMass) ∩
     (∀ targetsystembehavior TargetMass)))
   (∃ systemgoal (HeatSinkVolume ∩
     (= targetsystembehavior 1) ∩
     (∃ targetsystembehavior TargetVolume) ∩
     (∀ targetsystembehavior TargetVolume)))
   (≥ systemgoal 2)
```

```
(∃ objective.(HeatSink1DeviationFunction ∩
  (∃ function_of.( HeatSinkVolumeDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.HeatSinkVolume) ∩
     (∃ function_of.TargetVolume) ∩
     (∀ function_of.(HeatSinkVolume ∪TargetVolume)))
  (∃ function_of.(HeatSinkMassDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.HeatSinkMass) ∩
     (∃ function_of.TargetMass) ∩
     (∀ function_of.(HeatSinkMass∪TargetMass)))
(∀ objective.(HeatSink2DeviationFunction ∩
  (∃ function_of.( HeatSinkVolumeDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.HeatSinkVolume) ∩
     (∃ function_of.TargetVolume) ∩
     (∀ function_of.(HeatSinkVolume ∪TargetVolume)))
  (∃ function_of.(HeatSinkMassDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.HeatSinkMass) ∩
     (∃ function_of.TargetMass) ∩
     (∀ function_of.(HeatSinkMass∪TargetMass)))

  (= objective 1)
  (∃ behavioralrequirement.minimum_stiffness_requirement)))
```

6.3.3 Heat Sink Design Problem Example 3 – Thermal and Structural Model

The third example of the heat sink design problem takes into account structural and thermal considerations. The design goals are to minimize volume, thermal resistance, and mass. The heat sink design problem is described in Table 6-6. The design problem is represented as a cDSP in Figure 6-25.

Table 6-6: Fin array heat sink design problem description – Example 3

- The system variables include: number of fins, fin length, and fin thickness
- The design parameters include: ambient air temperature, volumetric flow rate of the cooling air, material properties of the heat sink, material properties of the cooling air, the power dissipated from the CPU, the width of the heat sink, the depth of the heat sink, heat sink load
- The design objectives are (1) minimize thermal resistance, (2) minimize the mass of the heat sink, and (3) minimize the size (volume) of the heat sink
- The fin array heat sink must remain within specified volumetric keep-in space based on the footprint of the CPU 80 mm $\times$ 80 mm $\times$ 50 mm
- The vertical stiffness of the heat sink is specified to be greater than 5,250,000 N/m to ensure proper preload and adequate contact
- The mass of the heat sink should minimized to ensure that mechanical shock and vibration are minimized to the CPU, the target mass is 0.01 kg
- The heat sink must be able to withstand a total clamping load, distributed over two clamping clips of 300 N
- The maximum steady-state temperature in the heat sink must not exceed 70 degrees C (343 K).

GIVEN

Design parameters

- Heat sink geometry parameters
- Heat sink material properties
- Cooling fluid material properties
- Loading & boundary conditions

Disciplinary Analysis Models

- Heat transfer analysis models
  1. Thermal resistance
  2. Maximum temperature
  3. Convection heat transfer coefficient
- Fluid analysis models
  8. Reynolds number
  9. Hydraulic diameter
  10. Fluid velocity
- Structural analysis models
  4. Vertical stiffness
  5. Compressive Stress
  6. Euler Buckling
  7. Inelastic Buckling
- Packaging analysis models
  11. Heat sink mass
  12. Heat sink volume

FIND

System variables

- Number of fins
- Fin length
- Fin thickness

Deviation variables

- Deviation from target mass
- Deviation from target thermal resistance
- Deviation from target volume

SATISFY

System bounds

- Bounds on fin width, $0.0001m \le t_{fin} \le 0.0005m$
- Bounds on fin length, $0.01m \le l_{fin} \le 0.05m$
- Bounds on number of fins, $2 \le N \le 30$

System design requirements & constraints

- Chip temperature is below maximum temperature
- Heat sink stiffness greater than minimum stiffness

Analysis models constraints & limitations

- *Model limitations imposed by analysis models*

System Goals

- Minimize deviation of heat sink mass from target mass, $M_{Heatsink} = f\left(N, L, t, l_{fin}, \rho_{Heatsink}, W, t_{base}\right)$
- Minimize deviation of heat sink thermal resistance from target thermal resistance, $R_t = f\left(W, t, k, t_{base}, \overline{h}, l_{fin}, N, L\right)$
- Minimize deviation of heat sink volume from target volume, $Vol_{Heatsink} = f\left(W, L, t_{base}, l_{fin}\right)$

Decision constraints

- $d_i^+, d_i^- \ge 0$ & $d_i^+ \cdot d_i^- = 0$ for all design objectives

MINIMIZE

Deviation function - $Z = f\left(w_{mass} \cdot d_{mass}^- + w_{resistance} \cdot d_{resistance}^- + w_{volume} \cdot d_{volume}^-\right)$

Figure 6-25: Fin array heat sink cDSP - Example 3

A graphical illustration of the cantilever beam design problem 3 is presented in Figure 6-22.

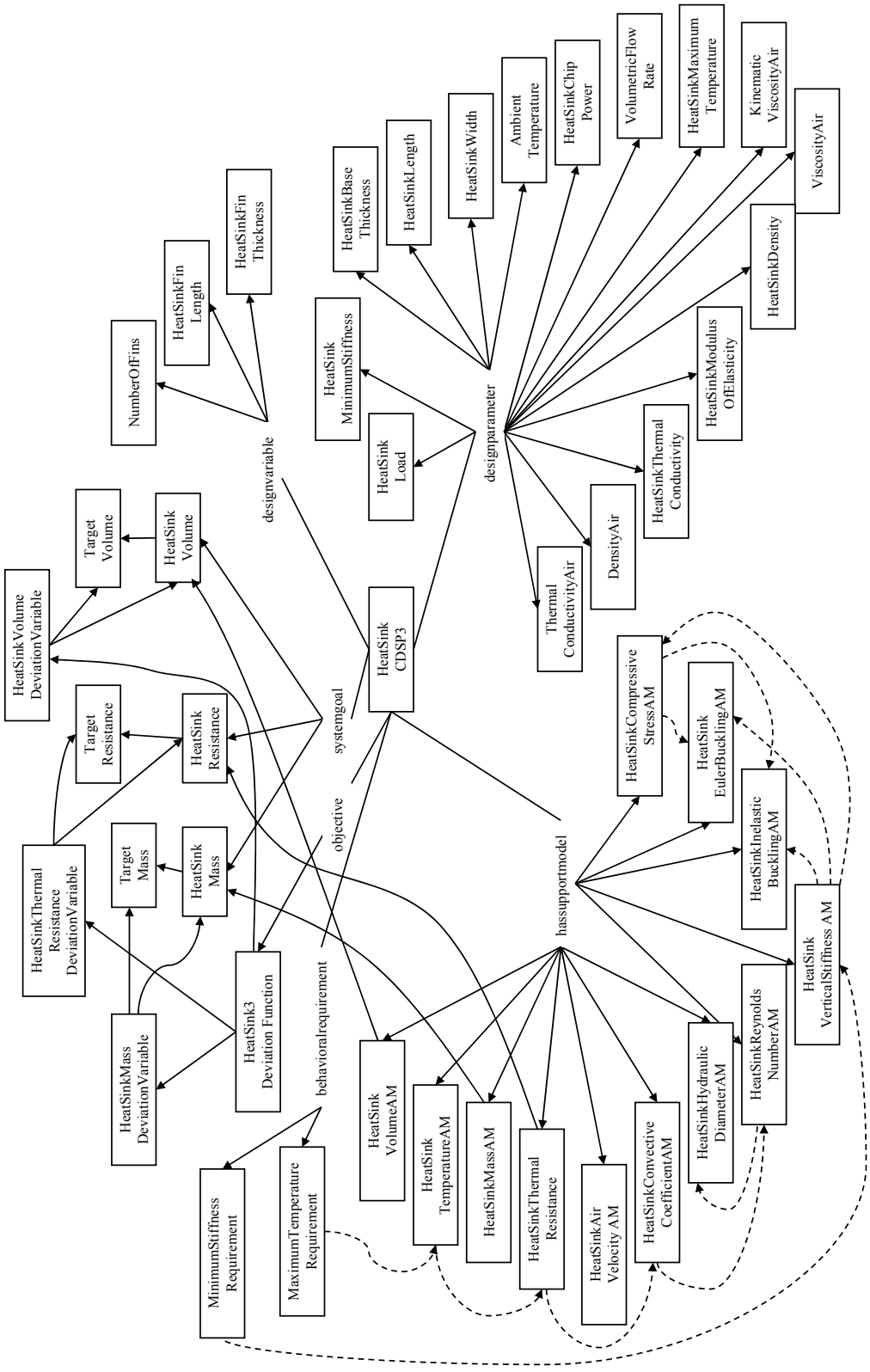


Figure 6-26: Graphical representation of heat sink design problem 3

The cDSP, illustrated in Figure 6-26, is a combination of the thermal heat sink design problem (Heat Sink Design Problem Example 1) and structural heat sink design problem (Heat Sink Design Problem Example 2). The heat sink example design problem 3 consists of 16 designparameters, three designvaraiables, three systemgoals, 12 analysis models, two behavioralrequirements, and a single deviation function. The coupling between the analysis models is illustrated in Figure 6-26 with dashed lines. For example, a coupling may exist between two analysis models (e.g., HeatSinkTemperatureAM and HeatSinkThermalResistanceAM) and a requirement and analysis model (e.g., HeatSinkTemperatureAM and MaximumTemperatureRequirement). As previously discussed, not all of the information is represented in the graphical model. For instance, the detailed relationships between the analysis model quantities and the design parameters and design variables are implied based on the analysis model concept definitions. The heat sink example design problem 3 is represented using the DL-based vocabulary as:

```
Class (HeatSinkCDSP3
and(
   (∃ designvariable.NumberOfFins)
   (∃ designvariable.HeatSinkFinLength)
   (∃ designvariable.HeatSinkFinThickness)
   (∃ designparameter.ThermalConductivityAir)
   (∃ designparameter.DensityAir)
   (∃ designparameter.HeatSinkThermalConductivity)
   (∃ designparameter.HeatSinkModulusOfElasticity)
   (∃ designparameter.HeatSinkDensity)
   (∃ designparameter.ViscosityAir)
   (∃ designparameter.KinematicViscosityAir)
   (∃ designparameter.HeatSinkMaximumTemperature)
   (∃ designparameter HeatSinkMinimumStiffness)
   (∃ designparameter HeatSinkLoad)
   (∃ designparameter.VolumetricFlowRate)
   (∃ designparameter.HeatSinkChipPpower)
   (∃ designparameter.AmbientTemperature)
```

```
 (∃ designparameter.HeatSinkWidth)
 (∃ designparameter.HeatSinkLength)
 (∃ designparameter.HeatSinkBaseThickness)

 (∃ hassupportmodel.HeatSinkVolumeAM)
 (∃ hassupportmodel.HeatSinkTemperatureAM)
 (∃ hassupportmodel.HeatSinkThermalResistanceAM)
 (∃ hassupportmodel.HeatSinkAirVelocityAM)
 (∃ hassupportmodel.HeatSinkConvectiveCoefficientAM)
 (∃ hassupportmodel.HeatSinkHydraulicDiameterAM)
 (∃ hassupportmodel.HeatSinkReynoldsNumberAM)
 (∃ hassupportmodel.HeatSinkMassAM)
 (∃ hassupportmodel HeatSinkCompressiveStressAM)
 (∃ hassupportmodel HeatSinkVerticalStiffnessAM)
 (∃ hassupportmodel HeatSinkInelasticBucklingAM)
 (∃ hassupportmodel HeatSinkEulerBucklingAM)

 (∃ systemgoal.(HeatSinkVolume ∩
   (= targetsystembehavior.1) ∩
   (∃ targetsystembehavior.TargetVolume) ∩
   (∀ targetsystembehavior.TargetVolume)))

 (∃ systemgoal.(HeatSinkMass ∩
   (= targetsystembehavior 1) ∩
   (∃ targetsystembehavior.TargetMass) ∩
   (∀ targetsystembehavior.TargetMass)))

 (∃ systemgoal.(HeatSinkThermalResistance ∩
   (= targetsystembehavior 1) ∩
   (∃ targetsystembehavior.TargetThermalResistance) ∩
   (∀ targetsystembehavior.TargetThermalResistance)))
 (≥ systemgoal 3))

(∃ objective.(HeatSink1DeviationFunction ∩
  (∃ function_of.( HeatSinkVolumeDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.HeatSinkVolume) ∩
     (∃ function_of.TargetVolume) ∩
     (∀ function_of.(HeatSinkVolume ∪TargetVolume)))
  (∃ function_of.(HeatSinkMassDeviationVariable ∩
     ((∃ relativeimportance.RelativeImportance) ∩
     (∀ relativeimportance.RelativeImportance)∩
     (= relativeimportance 1)) ∩
     ((∃ function_of.HeatSinkMass) ∩
     (∃ function_of.TargetMass) ∩
     (∀ function_of.(HeatSinkMass∪TargetMass)))
```

```
(∃ function_of.(HeatSinkThermalResistanceDeviationVariable ∩
      ((∃ relativeimportance.RelativeImportance) ∩
      (∀ relativeimportance.RelativeImportance)∩
      (= relativeimportance 1)) ∩
      ((∃ function_of.HeatSinkThermalResistanc) ∩
      (∃ function_of.TargetThermalResistance) ∩
      (∀ function_of.(HeatSinkThermalResistance∪
            TargetThermalResistance))))
  (∀ objective.(HeatSink1DeviationFunction ∩
    (∃ function_of.( HeatSinkVolumeDeviationVariable ∩
      ((∃ relativeimportance.RelativeImportance) ∩
      (∀ relativeimportance.RelativeImportance)∩
      (= relativeimportance 1)) ∩
      ((∃ function_of.HeatSinkVolume) ∩
      (∃ function_of.TargetVolume) ∩
      (∀ function_of.(HeatSinkVolume ∪TargetVolume)))
    (∃ function_of.(HeatSinkMassDeviationVariable ∩
      ((∃ relativeimportance.RelativeImportance) ∩
      (∀ relativeimportance.RelativeImportance)∩
      (= relativeimportance 1)) ∩
      ((∃ function_of.HeatSinkMass) ∩
      (∃ function_of.TargetMass) ∩
      (∀ function_of.(HeatSinkMass∪TargetMass)))
    (∃ function_of.(HeatSinkThermalResistanceDeviationVariable ∩
      ((∃ relativeimportance.RelativeImportance) ∩
      (∀ relativeimportance.RelativeImportance)∩
      (= relativeimportance 1)) ∩
      ((∃ function_of.HeatSinkThermalResistanc) ∩
      (∃ function_of.TargetThermalResistance) ∩
      (∀ function_of.(HeatSinkThermalResistance∪
      TargetThermalResistance))))
   (= objective 1)
(∃ behavioralrequirement.maximum_temperature_requirement)
(∃ behavioralrequirement.minimum_stiffness_requirement)))
```

In the previous sections, the declarative information associated with several analysis models and design problems are represented using a graphical notation and a DL-based computational representation. However, the cDSP represented using the formal language can not be solved. Thus, a manual "translation" process is required to develop executable decision representations. The analysis models and design decisions are implemented in

MATLAB and executed. In the following section, solution results obtained from the Heat Sink Design Problem 3 are presented.

6.4 Solving the cDSP for the Heat Sink Design Problem 3

The structural heat sink design problem 3 is similar to the cantilever beam design problem. An exhaustive search solution technique is utilized as a means for exploring the entire design space, eliminating the solution-dependency on specified starting values, and visualizing the design and analysis spaces for the beam problem. The exhaustive search is codified in MATLAB. The code for the heat sink is presented in Appendix B. The results from the design decision are presented in Table 6-7 and Figure 5-3. In this context, analysis meta-constraints define the space over which the analysis model produces valid results.

Table 6-7: Heat sink beam decision problem results

No analysis constraints						
$w_{Resistance}$	w_{Space}	w_{Mass}	l (m)	t (m)	N	Deviation Function
0	0	1	0.01	0.0001	2	0.90402
1	0	0	0.05	0.0005	30	0.3212
0	1	0	0.01	0.0001	2	0.90234
0.33	0.33	0.33	0.05	0.0005	30	0.74438
0.5	0.25	0.25	0.05	0.0005	30	0.64422
0.25	0.5	0.25	0.05	0.0005	30	0.80695
0.25	0.25	0.5	0.05	0.0005	30	0.80452
Analysis constraint considered						
$w_{Resistance}$	w_{Space}	w_{Mass}	l (m)	t (m)	N	Deviation Function
0	0	1	0.02	0.0002	28	0.92022
1	0	0	0.05	0.0005	30	0.3212
0	1	0	0.01	0.0004	29	0.92357
0.33	0.33	0.33	0.05	0.0005	30	0.74438
0.5	0.25	0.25	0.05	0.0005	30	0.64422
0.25	0.5	0.25	0.05	0.0005	30	0.80695
0.25	0.25	0.5	0.05	0.0005	30	0.80452

The results presented in Table 6-7 capture the different solutions when the analysis meta-constraints are considered. Graphical representations of the validity space for the heat sink design example are presented in Figure 6-27 and Figure 6-28.

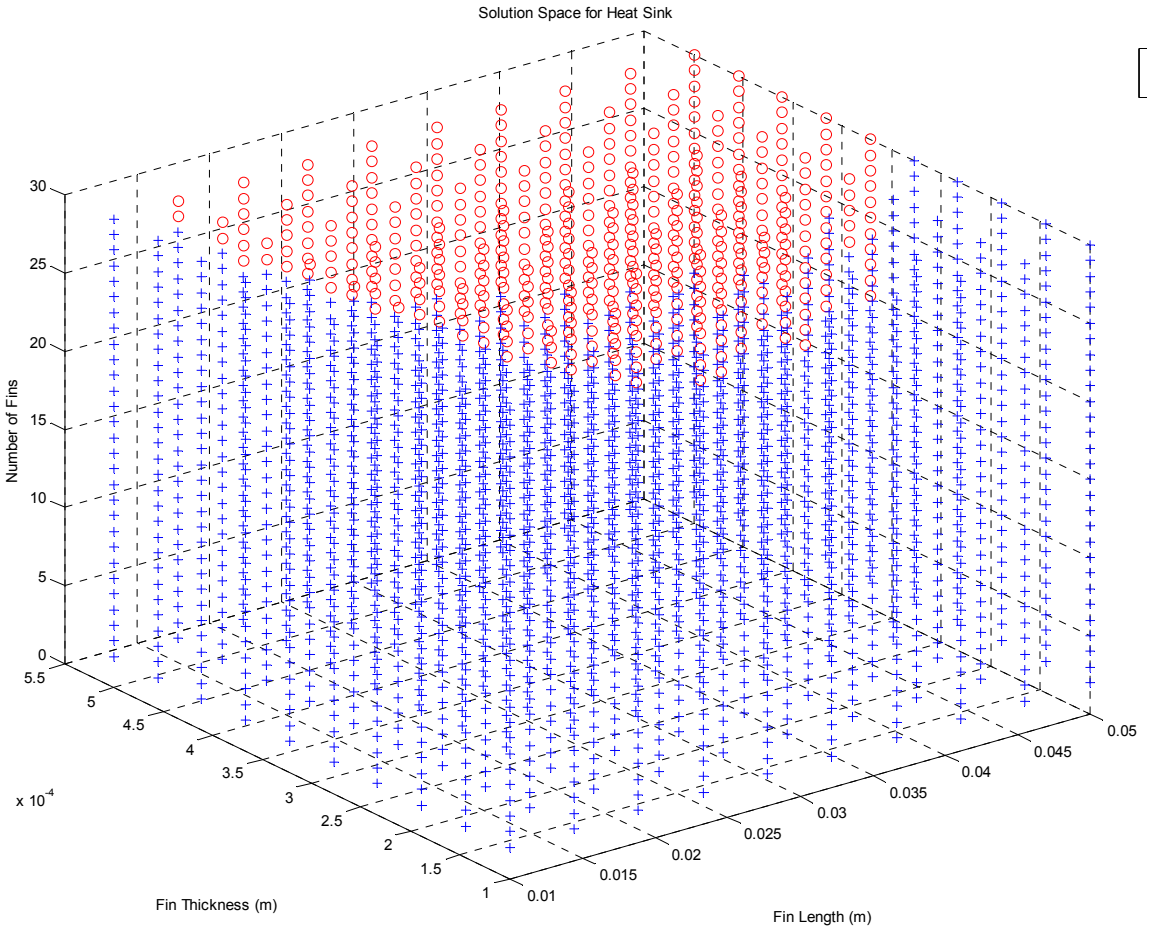


Figure 6-27: Validity space of heat sink design problem (w_{Mass} , $w_{thermal\ resistance}$, w_{space} = 0.33)

The red o's represent the valid design solutions and the blue +'s represent the invalid design solutions respectively. As illustrated in Figure 6-27, the design space is

considerably reduced when the analysis model constraints are considered in the design problem. The results from the solution can be interpreted in two ways. On the one hand, the analysis models can be thought of as too restrictive. Thus, higher-fidelity or different analysis models may be required to take advantage of the design space. For example, an alternative analysis model may have different assumptions and limitations, thus expanding the space over which the design solution is valid (i.e., the number of red o's is increased). On the other hand, the constraints imposed by the analysis model result in design solutions that are valid. Thus, the solution to the design problem can be accredited because the space over which the analysis model is valid is explicitly considered. Clearly, the constraints captured explicitly using the formal language dramatically affect the solution to the design problem.

However, to gain an even deeper understanding of the mechanism for producing a valid design solution, the space is further decomposed into four different spaces that reflect which constraint is violated. For example, the red o's represent the design solutions that satisfy the constraints specified by the designer, but violate the limitations and assumptions imposed by the analysis models used to support the design decision. The pink *'s represent those solutions that are valid with respect to the design constraint, but violate analysis constraints. The blue +'s represent the solutions that violate the design constraints, but satisfy the analysis constraints. Finally, the green *'s represent the decision solutions that violate both the design constraints and the analysis constraints.

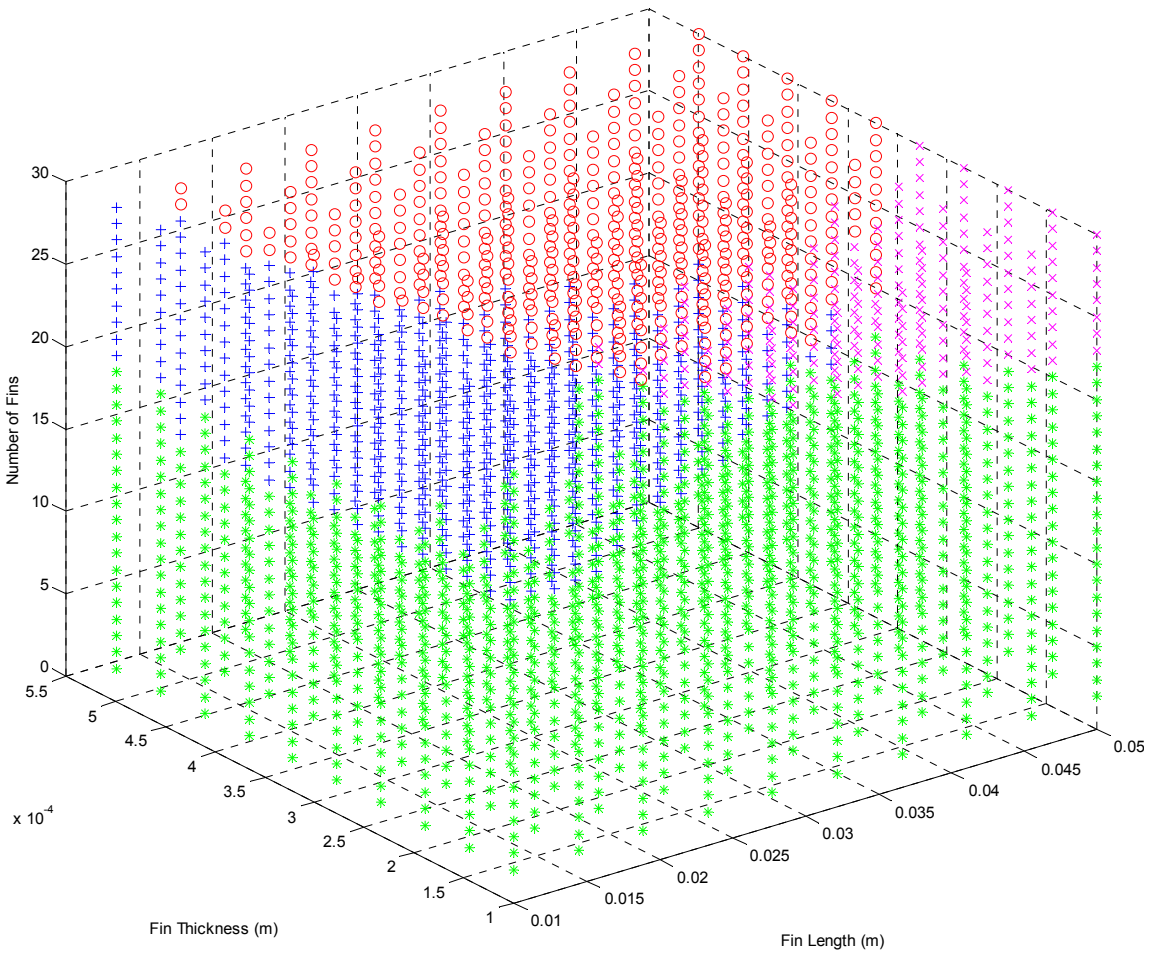


Figure 6-28: Analysis validity space of heat sink design problem

(w_{weight} , $w_{thermal\ resistance}$, w_{space} = 0.33)

Collectively, the pink *'s, blue +'s, and green *'s represent the invalid design solutions and are equivalent to the blue +'s presented in Figure 6-27. By visualizing the different regions of the design solutions, the design space can be more effectively explored. For example, if the design requirements are consistently violated, the designer may choose to evaluate the imposed design requirements. Similarly, if the analysis model limitations are violated, different models may be used to support the design decisions.

6.5 Discussion and Assessment of DL-Based Formal Language for the Heat Sink Design Problem

In the previous sections, the information associated with the design of a fin array heat sink is presented. The formal language developed in this research is used to explicitly capture the information associated with disciplinary analysis models and multi-disciplinary design decision. Specifically, graphical information models are developed to capture the complex information networks for thermal, fluid mechanics, and structural analysis models and multi-disciplinary cDSPs. Computational concept definitions are developed using DL based on the graphical information models. The vocabulary presented in Chapter 4 is used in conjunction with DL constructs (see Table 3-2) to developed complex concept definitions of the fin array heat sink design problems.

The same process discussed in Section 5.5, is followed for developing concept specifications for the heat sink design problem. The first step is the creation of specialized Quantity concepts (see Figure 6-29).

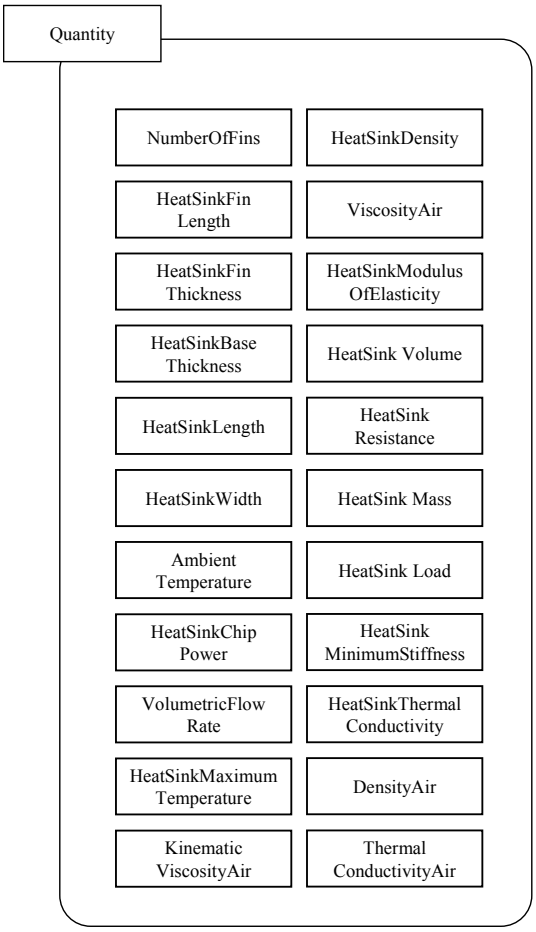


Figure 6-29: Specialized Quantity concepts associated with the heat sink design problem

As illustrated in Figure 6-29, there are 22 specialized Quantity concepts developed for capturing the heat sink design problem information. These quantities are used in conjunction with the cDSP vocabulary for developing complex concept definitions. The second step is specifying concept definitions for the ConstraintRelationship concepts. In this example, all of the ConstraintRelationhsip are implemented as EquationBasedConstraintRelationship concepts. Several constraint relationships are developed for representing analysis relationships, meta-level analysis relationship, and design requirement. The constraint relationship concept definitions are used for

specifying the information structure for disciplinary analysis models (see Section 6.2). The AnalysisModel and Quantity concepts are used in conjunction with DL constructs and the cDSP vocabulary for capturing the information associated with specific design decisions (see Figure 6-30.).

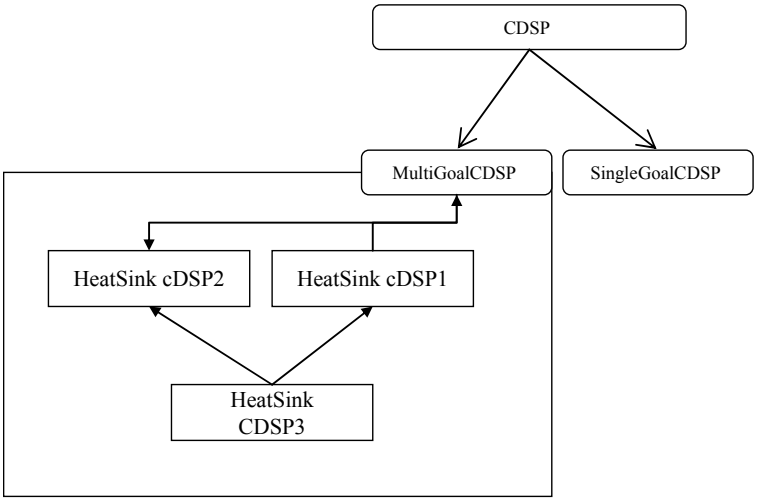


Figure 6-30: Hierarchical organization of heat sink decision model concept definitions

As illustrated in Figure 6-30, a hierarchical taxonomy of heat sink design decision is created based on DL reasoning algorithms. The heat sink example design problem 3 (HeatSinkCDSP3) is *implied* as a subclass of HeatSinkCDSP1 and HeatSinkCDSP2. This relationship is determined based on the concept specification and DL reasoning algorithms. As discussed in Chapters 4 and 5, the DL reasoning algorithms enable hierarchical information to be determined based on the concept specifications. In this example, the multi-disciplinary design decisions are organized in a hierarchical taxonomy based on the design variables, design parameters, and analysis models associated with the design decision. The DL reasoning algorithms facilitate the organization of decision-related information. A summary of the heat sink design problem is presented next:

Extensibility and robustness of the DL language - The cDSP vocabulary and DL representation have been illustrated to be extensible and robust by (1) developing specialized Quantity concepts for capturing specific design problems, (2) developing constraint relationships for capturing the analysis and metaanalysis relationships, and (3) concepts that represent specific design problems. The design of a heat sink is a multi-disciplinary problem that involves several analysis models. The DL language is successfully used for representing analysis models from several domains.

Consistency and organization of the heat sink design information - The consistency of the heat sink analysis models and design decisions is illustrated by developing a hierarchical taxonomy of engineering design decisions and analysis models based on the concept definitions. The organizational structure of the analysis model and cDSP is "up-to-date" through DL reasoning algorithms. As illustrated in Figure 6-30, the subclass relationships between the heat sink cDSP are determined based on the concept definitions of the heat sink cDSP. As new cDSP models are defined, the hierarchy is maintained.

Integration of analysis model information - The DL implementation facilitates the integration and exchange of information from multi-disciplinary analysis models. The heat sink design problem involves 12 analysis models that are coupled through the exchange of information. The DL-based implementation provides a mean for coupling analysis model in the context of the cDSP (see Figure 6-26).

The information associated with the heat sink design problems is successfully captured using DL. The DL-based implementation of the formal language provides a means for representing the semantics of design problems and analysis models in a computationally-interpretable language. Additionally, the decision-related information is

organized in a hierarchical taxonomy based on the concepts definitions and not explicit relationships defined between concepts.

6.6 Verification and Validation

In this chapter, the same validation and verification approach is taken as completed in Chapter 5. Confidence in the validity of the formal language is achieved by systematically increasing the complexity of and the aspects considered in the example problems. As previously stated, the formal language is utilized in Chapter 4 for representing the generic cDSP construct and analysis concepts. The representations in Chapter 4 lay the groundwork and the information structure for specific design problems. The information representation of the generic cDSP and analysis model concepts do not capture the number of quantities and analysis models associated with specific design problems. Thus, the complexity of the information representations is increased in Chapter 5 for a simple design problem. However, the information representations presented in Chapter 5 are limited to a single discipline, do not capture design problem requirements, and do not adequately represent the systematic formulation of design decisions. The examples presented in this chapter illustrate the applications and usage of the formal language for explicitly capturing the design parameter, variables, goals, constraints, and analysis models associated with the design of a cantilever beam. Furthermore, the several analysis model chains and coupling of physical phenomenon are present in the design problem. Finally, design and analysis requirements are captured in the cDSP representation.

Empirical structural validity (ESV) and empirical performance validity (EPV) are completed in this chapter by first evaluating the “appropriateness” of the heat sink

example problems, exercising the formal language (i.e., the vocabulary, information model, and DL implementation) for representing information associated with disciplinary analysis models and specific design problems, and completing the design decision and critically evaluating the results obtained. The verification and validation aspects completed in this chapter are summarized in Table 6-8.

Table 6-8: Validation and verification in Chapter 6

Empirical Structural Validation
§6.1 – The appropriateness of the heat sink design problems is established. An overview of the design problem is presented, followed by the detailed test plan and purpose of the examples in Table 6-1. The purpose of the heat sink example problems are presented in the context of addressing the research hypothesis. The fin array heat sink design problems are appropriate because the formal language can be used to demonstrate how to explicitly capture information associated with multi-disciplinary design problems and disciplinary analysis models. The example problems enable several aspects of the formal language to be demonstrated including the representation of design requirements, analysis model constraints, and hierarchical organization of design decisions.

Table 6-8: Validation and verification in Chapter 6 (continued)

Empirical Performance Validation
§6.2 – The formal language is utilized for explicitly capturing the information associated with disciplinary analysis models including structural analysis models, thermal analysis models, fluid mechanics analysis models, and general engineering analysis models. In addition, several analysis models are developed to simulate specific behavioral phenomenon such as stress, buckling, fluid flow, heat transfer coefficient to name a few. In addition to representing the behavioral relationships, the formal language is utilized for capturing analysis model constraints and limitations. The analysis models are organized in a hierarchical fashion based on the concept definitions by taking advantage of DL standard reasoning services.
§6.3 – Three design problems are represented as cDSP using the formal language. Graphical representations and DL implementations are realized for capturing the information associated with the design problems in a computational manner. Several aspects of the design problem are addressed using the formal language. First, the design and analysis requirements are represented in the cDSP as design constraints. The semantics and the mathematics of design requirements are represented using the constraint relationship concepts. Second, several disciplinary analysis models are integrated into the design problem. The analysis models are integrated through the digital interface for exchanging information. The digital interface is established using the formal language. Additionally, the graphical representation enables coupling between thermal analysis models, fluid mechanic analysis models, and structural analysis models to be visualized. The three design problems are similar, but differ in the behaviors considered and the design requirements modeled. Holistically, the design problems provide insight into the reuse, retrieval, and organization of decision-related design information. For example, the network of relationships between decision models is captured and explicitly represented through DL reasoning algorithms.

Table 6-8: Validation and verification in Chapter 6 (continued)

Empirical Performance Validation
§6.4 – The declarative information associated with the disciplinary analysis models and multi-disciplinary design decisions is "translated" into procedural code and subsequently executed using an exhaustive search solution approach. The results from decision models are discussed and related to the information representations. The results do not directly support the "correctness" of the formal language, but rather provide support evidence for explicitly representing decision related information. For example, the results indicate the importance of capturing analysis-related limitations and assumptions and the validity of the design decision solutions.

As previously stated, empirical structural validation (ESV) involves accepting the appropriateness of the example problems used to verify the performance of the method. In the case of this research, the example problems are used to demonstrate the correctness and completeness of the formal language. The example problems discussed in this chapter involve the multi-disciplinary design of fin array heat sink for electronic cooling applications. In the design of fin array heat sink several disciplines and analysis models must be integrated from multiple design disciplines. In the context of heat sink design, analysis models are integrated from fluid mechanics, heat transfer, and structural mechanics. In addition to integrating the models into the design decision, several models are coupled together through the sharing of design variables or behavioral variables. For example, the stress analysis model used to predict the compressive stress is a single heat sink fin is couple to the Euler buckling analysis model through an analysis constraint namely, "the compression analysis model is only valid when the compressive stress in the fin does not reach a critical buckling stress."

The design problems require the integration and exchange of decision-related information from multiple design perspectives. We believe the example problem is of reasonable complexity and enables several different aspects of the research questions to be addressed. Thus, the heat sink design problems are appropriate for the validation of the formal language.

Empirical performance validation (EPV) consists of accepting the usefulness of the outcome with respect to the initial purpose and accepting that the achieved usefulness is related to applying the method. The empirical performance validation in this chapter is carried out by explicitly representing the information associated with disciplinary analysis models in Section 6.2. The information associated with the disciplinary analysis models is represented using the graphical information model and in a computational means through the use of DL. Analysis relationships and meta-relationships are explicitly captured by creating specialized Quantity and ConstraintRelationship concepts that are used for representing information specific to analysis models and design decisions. The specialized Quantity concepts provide a mean through which disciplinary designers can communicate. Thus, information exchange is enabled through a commonalized vocabulary consisting of physical quantities. The disciplinary analysis models are integrated into multi-disciplinary design problem in the context of the compromise decision support problem. The formal language is used to demonstrate how the formal language enables the integration of multiple design discipline into a unified decision construct. Additionally, the exchange of information in addition to analysis results is illustrated. For example, the formal language enables analysis model constraints to be captured and shared in multi-objective design problems. Finally, the declarative

information representations are implemented in procedural code and solved. The results obtained from the design decisions are discussed in the context of the formal language. For example, the actual results obtained for the solution are not of central importance rather the insight gained into the effects of the information captured using the formal language are discussed. For example, the representation of analysis model limitations has a profound effect the validity of the design solutions and therefore must be captured.

The heat sink design problems have demonstrated the value of the formal language for representing decision-related design information. The formal language is shown to be effective and complete in the context of modeling engineering decisions as compromise decision support problems. The formal language enables complex decision and analysis information to be represented using an established vocabulary and limited set of DL constructs. Reasoning and retrieval is support using the decision vocabulary and the concept definitions developed using description logic. It is shown in Sections 6.2 – 6.3 that he formal language enables digital interfaces to be established using a relatively small number of unique concepts, thus enabling information exchange across multiple design disciplines. In Section 6.2, the information associated with 12 analysis models is captured. The analysis relationships and meta-level constraints are represented using the base vocabulary. The analysis models are "published" to the information model as they are specified. In Section 6.3, three design problems are modeled and the information is represented using the formal language. Like the analysis models, the cDSPs are published as they are specified and dynamically organized in a hierarchical fashion. The integration of analysis models and engineering design decision is illustrated in Section 6.3. Linkages are established between analysis models and the associated quantities and design

decisions. As previously stated, the formal language provides a structured, computationally-based approach for capturing the information associated with multi-objective design decisions. Based on the heat sink design problems discussed in this chapter, it is argued that the formal language is appropriate for capturing the information associated with deign problems. Additionally, the language us complete in the context of modeling disciplinary analysis models and multi-objective, multi-disciplinary design decisions.

6.7 Chapter Synopsis

In this chapter, the design of fin array heat sink is presented as an example problem for demonstrating the use of the formal language for capturing and explicitly representing information associated with multi-objective design decisions (see Figure 6-31).

- ✓ To demonstrate the use of the formal language for a complex, multi-disciplinary design problem - The information associated with three heat sink design problems is captured including 1) a thermal design problem, 2) a structural design problem, and 3) a thermal-structural design problem. The information associated with several disciplinary analysis models are represented including: thermal, structural, and fluid mechanics.
- ✓ To explicitly capture the relationships, both analysis and meta-level, for disciplinary analysis models – The information associated with engineering analysis models and the assumptions and limitations are captured. Several analysis models are represented using the formal language and the meta-level constraints the define the validity space of the analysis models are captured.

- ✓ To illustrate the importance of capturing design information to enable integration and exchange – The importance of capturing richer representations of decision and analysis information is demonstrated by solving the design decisions and discussing the results. The solutions to the design problems are drastically different and often time invalid when the assumptions and limitations of the analysis support models are not considered. Thus, several cDSP are developed in which the assumptions and limitations of the analysis models are propagated to the design decision, ultimately effecting the design outcome.

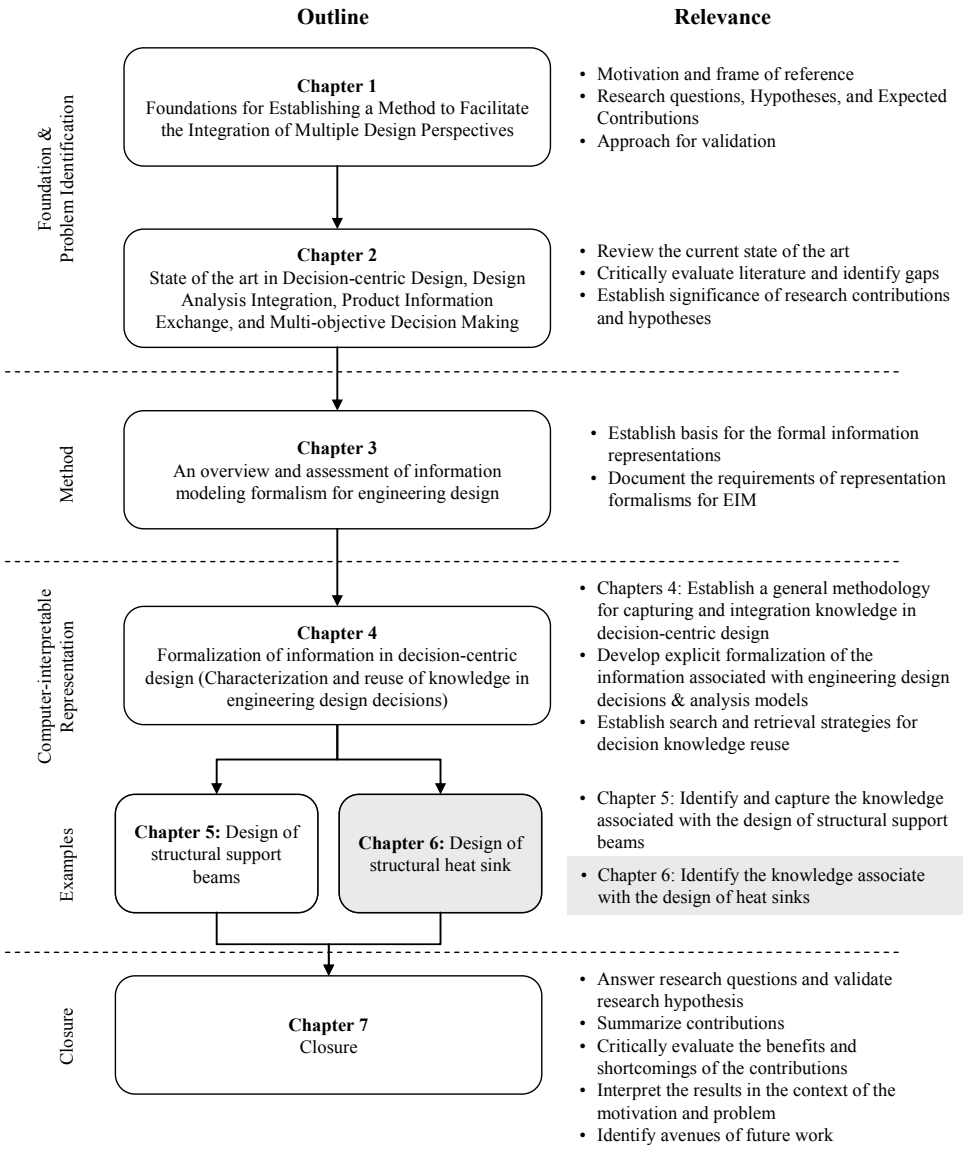


Figure 6-31: Outline of dissertation

The fin array heat sink design examples presented in this chapter provide additional support, both qualitative and quantitative, for verifying and validating the hypotheses posed in this research. The heat sink design problems are used to demonstrate the formal language for explicitly capturing decision related information and enabling the integration of disciplinary analysis models. In this example, several analysis models are represented and used to support a design decisions. The graphical information model and

DL representation are used to gain insight about the information network associated with design decisions. In addition, the declarative information representations realized through the formal language are represented as executable representation, which are subsequently solved using traditional optimization techniques. The solution results obtained from executing the design decision illustrate the importance of explicitly capturing decision related information, and thus indirectly the importance of the formal language.

The following chapter is the closure of this dissertation. A holistic discussion is presented based on the examples from Chapters 4, 5, and 6. A summary of the dissertation is presented of the first six chapters as well as a discussion of the validation, research contributions, and avenues for future work.

CHAPTER 7:

CLOSURE

The development and presentation of the information model are brought to a close in this chapter. The research questions raised at the beginning of this dissertation are discussed in the context of the validation approach. More importantly, the research hypotheses posed in Chapter 1 are evaluated in the context of the research findings. In this chapter, the validity of the three hypotheses are argued to build confidence in the principle hypothesis and overarching research objective. Finally, a critical review of the limitations and future research directions are discussed.

7.1 Summary of Dissertation

In this dissertation, a formal language for capturing the semantics of multi-objective engineering design decisions and analysis support models is developed. The formal language developed in this research is an integral component for communicating and exchange product-related design information in distributed, collaborative product development. The formal language is established as a realization of the primary research hypothesis posed in this dissertation: "Description Logic-based information models will provide a formal language for integrating multi-disciplinary decision knowledge."

The primary hypothesis is formulated based on the primary research question addressed: "How can information from multiple sources be (a) systematically captured and (b) formally represented in a computational means to facilitate integration in decision-centric design?" The formal language developed in this research consists of

three components: (1) a systematic method for formulating multi-objective engineering design decisions. The method is comprised of seven phases that are divided into two sub-methods. The first sub-method is focused on capturing decision related information and the second sub-method is focused on characterizing disciplinary analysis models for reuse. The method provides a structure framework for capturing decision related knowledge; (2) a graphical information model for capturing the semantic of engineering design decisions. The information enables decision related information to be captured in a structured manner in accordance with the systematic design method. The information models serves as the foundation for developing computational implementations; and (3) a computer-interpretable representation of engineering decision information, implemented in Description Logic, is proposed as a means for capturing decision information. The DL implementation enables designers to create description of design problem using an established vocabulary. The knowledge representation serves as a "common language" through which designers from multiple perspectives can integrate knowledge and formulate design decisions. There are three important aspects of the formal language that must be addressed. The primary research question is decomposed into four, closely related sub-questions that address the three components of the formal language.

- RQ1a: How can the structure and semantics of the compromise decision support problem be captured?
- RQ1b: How can the structure and semantics of analysis support models be captured to facilitate integration in the cDSP?
- RQ2: How can the information and knowledge associated with cDSP and analysis support models be represented in a computational environment?

- RQ3: How can cDSP-related knowledge be organized and retrieved to enable reuse?

The research questions are posed in the context of addressing the underlying problem associated with information exchange in engineering design decisions: "*Engineering designers make decisions based on information from multiple sources that span various perspectives in the product realization process. However, the information is often independent, limited to a single perspective, and not formally represented making it difficult to exchange and share in the context of engineering design decisions."*

The context for answering the aforementioned research question and overarching problem addressed in this dissertation is to facilitate communication between design perspectives. The principle goal in this research is to d*evelop a computational knowledge representation (i.e., formal language) for capturing the semantics of engineering design decisions to facilitate the integration of knowledge from multiple design perspectives.* Multi-objective compromise decision support problems are information intensive constructs that require the integration of information from multiple perspectives into a unified decision model. For example, cDSPs are characterized by the integration and exchange of information between diverse design perspectives including: 1) sharing and exchanging design variables and parameters between diverse perspectives, 2) chaining and coupling of disciplinary analysis models, and 3) imposing analysis model constraints on the solution space. Hence, the explicit representation for capturing decision-related information is addressed throughout this dissertation with a focus on the development of a formal language based on DLs.

In this dissertation a formal language, comprised of a graphical information model, vocabulary, and DL implementation, is developed and used for explicitly representing the

information associated with cDSP and analysis support models. The formal language provide a computationally-based approach for capturing the semantics associated with engineering design decisions and provides a link between the abstract design problem and the mathematical optimization problem. The formal language requires an predetermined vocabulary for describing the concepts and properties within a particular domain and a grammar based on description logics for developing complex definitions.

From a big picture perspective, the formal language addresses each of the research questions. The research questions are closely related and build on each other to address the overarching research goal in a systematic manner. First, a systematic method is developed as a means for structuring the process of formulating cDSPs. The focus of the systematic method is identifying the information generated and utilized for a design decision. The outcome of the systematic method is a mathematical-based notation that captures the decision-related information associated with the cDSP construct and supporting analysis models. The systematic method results in a well-defined set of steps and phases and an established set of information associated with the mathematical form of the cDSP. The systematic method partially addresses the first research question (RQ1a & RQ1b).

Second, a vocabulary is extracted from the systematic method. The vocabulary consists of concepts and properties (i.e., relationships between concepts) that are used to build definitions for capturing the semantics associated with cDSPs. The final vocabulary is settled on as a result of an iterative development process in which the criteria established by Gruber [60] is used for assessing the "goodness" of the vocabulary. The vocabulary provides a set of basic definitions through which additional concepts are

defined. The vocabulary addresses the first and second research questions. Third, the graphical information model is developed to address the first research question (RQ1a & RQ1b). The graphical representation is used to explicitly model the information associated with the cDSP and associated analysis support models from a conceptual perspective. The graphical representation is similar to the modeling approaches utilized in database design and development. The graphical information model consists of nodes that represent concepts and links that represent relationship between concepts.

The second and third research questions are addressed through the development and implementation of DL representations of cDSP and engineering analysis models. The vocabulary and graphical information models are used to explicitly capture the information associated with cDSP constructs. However they are not computer-processible. Hence, a computation-based language is required. Description Logic is utilized as a means for developing the formal decision language (i.e., a computer-processible representation of a domain of discourse).

The approach and formal language developed in this research are demonstrated and validated through several example problems including the generic cDSP, single-goal cDSP, multi-goal cDSP, Type I and Type II Robust cDSP, generic analysis model representation, computational-based and equation-based analysis models. Additionally, the formal language is demonstrated through several example design problems including the structural design of a cantilever beam, and multi-disciplinary design of a structural fin array heat sink for electronic cooling. Validation and verification of the formal language is achieved by systematically increasing the complexity of and the aspects considered in the example problems. First, the formal language is utilized in Chapter 4 for representing

the core concept definitions associated with multi-objective decision making. The cantilever beam, presented in Chapter 5, is a simple example that involves a single discipline (i.e., structural mechanics) and is used to demonstrate the use of the formal language for explicitly capturing decision and analysis related design information. The heat sink design example is a complex example that involves the integration of several disciplines including fluid mechanics, heat transfer, and structural mechanics. In Chapter 6, the heat sink design example is used to validate the formal language for an increasingly complex example. A detailed discussion of the examples presented in this dissertation is discussed in the context of validation and verification in Section 7.2.

In summary, the primary contribution from the work completed in this dissertation is a formal approach for supporting distributed, collaborative product development by enabling effective communication in multi-disciplinary design decision making. Specifically, the contributions in this research include a) a method for systematically formulating and integrating disciplinary information for multi-objective design decisions, b) a graphical information model for capturing the semantics of engineering design decisions, and c) a computer-interpretable representation of engineering decision information using DL. The details of the contributions are discussed in Section 7.3.

7.2 Answering the Research Questions and Validating the Hypothesis

As stated in Chapter 1, the goal of this research is to develop a formal language for capturing the semantics of engineering design decisions. The goal in completing this research is to provide the foundation for developing a formal language for exchanging and integrating information for multi-objective decision making (see Figure 7-1).

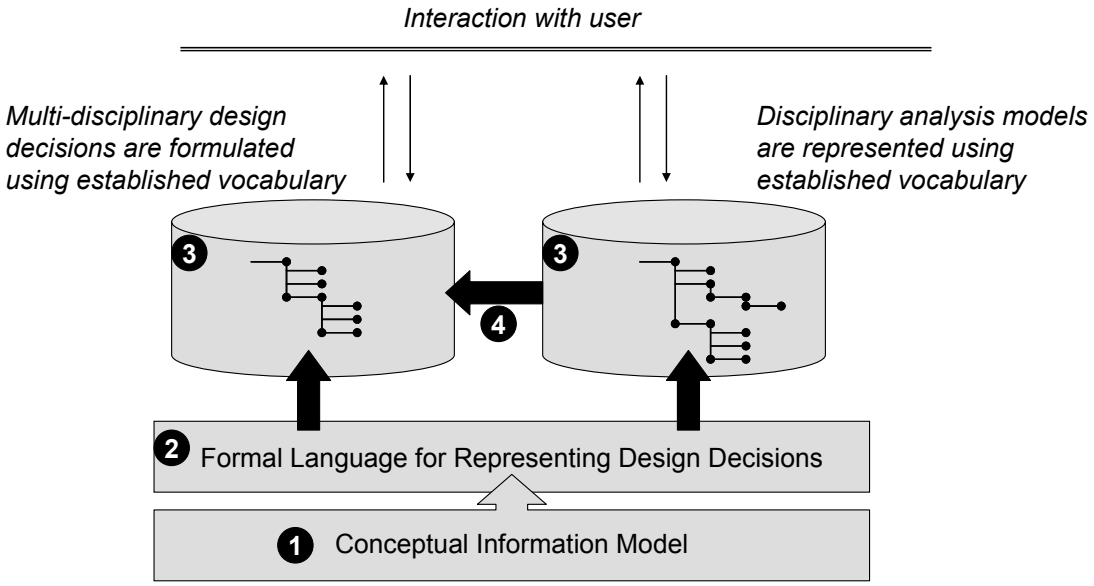


Figure 7-1: Envisioned framework for representing decision information

As illustrated in Figure 7-1, the envisioned framework consists of: ❶ a conceptual information model for capturing the semantics of engineering design decisions (i.e., the vocabulary and graphical information model); (❷❸) a computer-interpretable formal language based on the conceptual information model and implemented using description logic; and ❹ algorithms to support retrieval and organization of decision-related design information using standards reasoning supported by DL. Hence, the primary objective in this research is to d*evelop a computational knowledge representation (i.e., formal language) for capturing the semantics of engineering design decisions to facilitate the integration of knowledge from multiple design perspectives.* To realize this objective, the primary research question must be answered: *How can information and knowledge from multiple sources be (a) systematically captured and (b) formally represented in a computational means to facilitate the integration of multiple perspectives in the context of*

a decision-centric product realization process? The primary research question is an attention directing tools for formulating the primary hypothesis. The primary hypothesis is: *Information and knowledge from multiple design perspective and domains can be integrated by developing computer-interpretable representations of decision information.* The primary research and hypothesis are broken down into a number of supporting questions and hypotheses. A detailed discussion and summary of the validation and verification considerations for each of the sub-research questions and hypotheses are presented in Section 7.2.1, 7.2.2, and 7.2.3. A discussion regarding the theoretical performance validity of the research questions and hypotheses is presented in Section 7.2.4 which involves systematically building confidence in the formal language through the examples presented in this dissertation for design scenarios beyond what is explicitly shown. Theoretical performance validity addresses the notion of *generality* of the contributions from this dissertation.

7.2.1 Sub-Research Question 1 - Structure and Semantics of Decision-Related Information

The first research question further decomposed into two closely related research question that share a common hypothesis: *"RQ1a: How can the structure and semantics of the compromise decision support problem be captured?"* and *"RQ1b: How can the structure and semantics of analysis support models be captured to facilitate integration in the cDSP?"* The answer to this question is supported by the following hypothesis: *"Hypothesis 1: Information models can be developed to explicitly represent the concepts, and properties of concepts associated with engineering design decisions and analysis support models".*

In Hypothesis 1, we proposed that an information model could be developed as a common and structured vocabulary for describing design decision and analysis models. To validate this hypothesis we have developed several information models for representing a variety of design decisions and analysis models to test the expressiveness and robustness of the model. The information models are developed based on established design decision constructs, namely the compromise decision support problem and multi-objective design optimization formulations, and several analysis support models. Additionally, the information modes are developed using information an established, well-accepted ontology modeling framework [88; 156]. The framework consists of four phases in which (1) the high level requirements are identified for selecting a particular modeling formalism, (2) the scope of the information model is clearly defined by identified the requirement for the model, (3) the information model is developed and implemented based the particular modeling formalism that is selected and the scope of the information model, and (4) the information model is validated and checked to ensure the correctness and completeness based on the high-level modeling requirement and requirements the define the scope of the model (see Figure 7-2).

The information model is developed using a graphical representation similar to the entity relationship (ER) model in which concepts (entities) and properties (relationships) are represented. Entity relationship and database method have been widely used in business process and are gaining momentum in engineering, thus we believe this approach is valid. The ER was not directly used because of the decision to use description logic (DL) as the representational formalism. In this context, DL is limited to properties between two concepts, where as ER enable increased cardinality relationship. A

simplified graphical representation that represents concepts and binary properties is used for explicitly modeling engineering design decision and analysis support models.

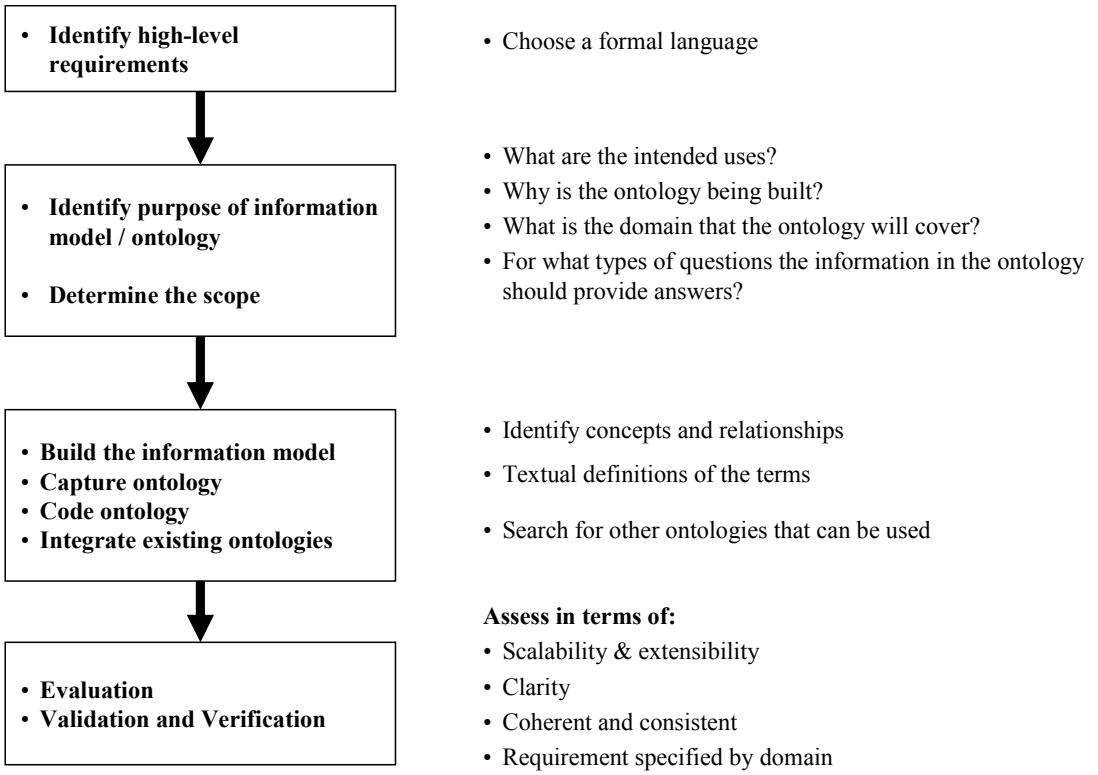


Figure 7-2: Information modeling framework

The graphical representation is used to capture the concept and data-type properties between concepts (see Figure 7-3).

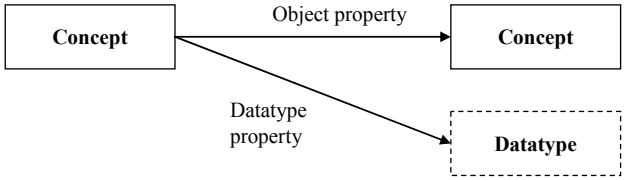


Figure 7-3: Graphical representation for decision information

Specifically, representations are created that explicitly model the information associated with several different design decision formulations including the cDSP and

robust design decisions. Additionally, the information model is assessed in terms of several criteria put forth by Gruber [60] and the requirement for modeling engineering information management problems.

The Theoretical Structural Validity (TSV) of Sub-Research Questions 1a and 1b and Hypothesis 1 is completed by establishing the internal consistency of the cDSP. The internal consistency of the cDSP is established by conducting and critically evaluating existing literature in the domains of information modeling, decision-centric design, and the compromise decision support problem and decision support problem technique. Based on existing literature it was determined that the formal representation of decision-related information is of central importance for integrating multiple design perspectives in the product realization process, but has not been adequately addressed. Information modeling efforts have predominantly focused on the representation of product-related information (i.e., geometry), while the research in design decision making has not addressed information modeling. Each of the research fields have developed well-accepted sound constructs, but have not been synergized. Based on the literature, a list of requirements is formulated for capturing the information associated with multi-objective engineering design decisions (§4.1), a systematic method is developed for capturing the information associated with the cDSP construct (§4.2), and a vocabulary for describing the semantics of design decisions is developed (§4.3). Hence, based on existing literature, Hypothesis 1 is appropriate and theoretical structural validation is completed.

Empirical validation of Hypothesis 1 is completed in two steps. The first step is to ensure the example problems developed in the research are appropriate and test sufficient

aspects of the hypothesis to gain confidence. The second step is exercise the examples to actually test those aspects of the hypothesis. Empirical structural validation is completed by systematically increasing the complexity of and the aspects considered in the example problems. Several examples are proposed to empirically validate the hypothesis including: the traditional cDSP construct, single- and multi-goal cDSP, generic analysis model, equation-based and computational analysis model, cantilever beam design problems, and heat sink design problems. The empirical structural validity of each of the example problems is established in an appropriate section. The empirical structural validation of the general cDSP and analysis models is discussed in Table 4-1 and Section 4.9, the cantilever beam design examples is discussed in Table 5-1 and Section 5.5, and the heat sink design examples is discussed in Table 6-1 and Section 6.5.

Empirical performance validation is completed by using the formal language to explicitly represent and integrate the information associated with each of the afore-mentioned examples. The formal language has been successfully exercised in Chapter 3, 4, and 5 for capturing the information associated with disciplinary analysis models and multi-disciplinary design decisions. Based on results from the example problems, we observe that the formal language provides a means for describing and capturing the semantics associated with design decision. The graphical information model provides a means for visualizing cDSP and analysis models information. Additionally, the graphical representation provides a means for identifying required information and coupled analysis models, and design constraints. More over, the vocabulary enables the information to be published using predetermined semantics, thus enabling integration and information exchange. The example problems included in this dissertation are appropriate

and all aspects of Hypothesis 1 are tested through the example problems. Hence, empirical structural validity and empirical performance validations is completed and our hypothesis is supported. Thus, information models can be developed using a base vocabulary for formally describing the information associated with the cDSP decision construct and associated support models to enable effective communication and collaboration.

While the formal language enables us to address Hypothesis 1 and provide several advantages, fundamental disadvantages exist that limit the applicability. An advantage of the formal language is the establishment of a computer-based representation that enable designers to communicate unambiguously in the context of the cDSP decision construct. However, a limitation of the graphical information model is the overhead required to actually capture the information diagram associated with complex models. As illustrated in Section 6.2 and 6.3, the number of concepts and links in the graphical representations quickly increases to the point where the representation becomes a "spaghetti" of concepts and relations, thus diminishing the value and insight gained from the representation. Additionally, the composition of several sub-graphs is maintained manually. This is both a limitations and an advantage. It is an advantage because the formulation of design decisions is not completely automated, keeping the responsibility of the designer In the loop. However, it is a disadvantage because a semi-automated or user-controlled automation would increase the computer support for decision formulation. The vocabulary and graphical representation are in the early stage of development and provide a foundation for developing search and composition algorithms as well as support tools. The limitation is discussed in further detail in the future work, Section 7.4.

7.2.2 Sub-Research Question 2 - Computer-Processible Formal Language Representation

The second research question is: *"How can the information associated with the cDSP and analysis support models are represented in a computational environment?"* It is essential that the information representation for engineering design decisions is computer-processible. The second research hypothesis addresses the need to develop computer interpretable information representations: *"Hypothesis 2: Description Logic can be used for realizing the information model in a computational means"*. This hypothesis is validated by critically evaluating information modeling formalisms in the context of engineering design problems in general and cDSPs in particular and using DL for representing the information associated with specific cDSPs and analysis models.

Theoretical structural validation is completed by evaluating several information modeling formalisms in the context of engineering design problems and completing a critical review of current literature of information modeling formalisms. Theoretical structural validity is established by determining the internal consistency of DL for addressing problem associated with engineering information modeling. First, a set of design characteristics and information modeling requirements are established by critically reviewing existing literature from the fields of information modeling, systematic design methodologies, and frameworks for integrating multiple design disciplines [67; 72; 112; 129; 164]. Three information modeling formalisms are evaluated in the context of the identified modeling requirements. Based on this evaluation, DL is recommended as the formalism for developing information models of engineering design decisions. The internal consistency of DL is determined by critically evaluating current research and

development efforts and applications of DL for information modeling. In Section 3.6.4, it is argued that DL is appropriate for modeling information associated with engineering design and offers advantages over other modeling formalisms. Based on existing literature, it is shown that DL has been used for conceptual information modeling in several domains including medical, configuration management, and database development [16]. It is determined that DL provides several advantages over other modeling formalism for addressing engineering information management problem. The advantages of DL are related to the mathematical properties and performance of DL. For example, Baader and colleagues [16] discuss the internal consistency and complexity of DL in depth. Thus, the mathematical properties of DL provide support for internal consistency because of the completeness and soundness of reasoning algorithms. Thus, theoretical structural validity of Hypothesis 2 is partially completed. However the use of DL, up to this point, has not been discussed in the context of information modeling of engineering decisions. Consequently, Research Question 2 and Hypothesis 2 are not fully validated. As discussed in Section 7.2.1 the theoretical structural validity is established for the cDSP and analysis models. Thus, to fully validate and verify Hypothesis 2, the vocabulary developed in Hypothesis 1 and description logic are discussed collectively. The cDSP vocabulary is comprised of concepts and properties for modeling the semantics of decisions explicitly. In general, DL provides a predetermined set of construct (i.e., grammar) that is used in conjunction with concept and properties to create complex definitions for a domain in a computer-processible representation. A logical procedure is followed to establish the internal consistency of the cDSP and DL for engineering information modeling. The vocabulary is a common characteristic that provides a link

between the domain of discourse (i.e., cDSP) and the information modeling formalism (i.e., DL). Due to the logical procedure of establishing the theoretical structural validity and the common characteristics, the theoretical structural validity of Research Question 2 and Hypothesis 2 is established.

Empirical validation of Hypothesis 2 is completed by first establishing the appropriateness of the examples and then testing the examples in the context of Research Question 2 and Hypothesis 2. The same examples are utilized to verify and validate all of the hypotheses in this research. Thus, because the empirical structural validity of the example problems is established for Hypothesis 1 and Hypothesis 1 and 2 are closely related, we accept the empirical structural validity of the example problems for Hypothesis 2. Specifically, the example problems enable the robustness and completeness of the formal language to be tested, the organization and consistency of DL reasoning algorithms to be tested, and the representation and integration of multi-disciplinary design decisions and disciplinary analysis models to be tested. Additionally, the example problems provide a testbed to evaluate the formal language in accordance with the criteria established by Gruber [60]. The example problems range in complexity and are realistic. The empirical structural validation of the general cDSP and analysis models is discussed in Tables 4-1 and 4-12 and Section 4.9, the cantilever beam design examples is discussed in Tables 5-1 and 5-5 and Section 5.5, and the heat sink design examples is discussed in Tables 6-1 and 6-8 and Section 6.5.

Empirical performance validation for Hypothesis 2 is completed by developing DL-based representations of the language for explicitly representing and integrating the information associated with each of the afore-mentioned examples. The DL

representation is used successfully in Chapter 4, 5, and 6 for capturing the information associated with disciplinary analysis models and multi-disciplinary design decisions. For example, DL is used for specifying the information structure associated with the cDSP and analysis models in Sections 4.4 through 4.6. Similarly, DL representations of the cantilever beam disciplinary analysis models and multi-disciplinary design decisions are developed in Sections 5.2 and 5.3. Finally, the heat sink design decisions and analysis models are represented in Sections 6.2 and 6.3. Based on the example problems, we observe that the DL representation provides a means for describing and capturing the semantics associated with design decision in a computationally processible means. Additionally, the Protégé-OWL development environment is used for developing DL-based representations of the decisions and analysis models associated with the cantilever beam and heat sink design problems. Hence, empirical validity of the DL representation is accepted for Hypothesis 2.

In critically assessing the DL representation developed to address Hypothesis 2, it is important to identify the underlying advantages and limitations. The advantages and limitations of the DL implementation are roughly categorized into theoretical advantages limitations and implementation advantages and limitations. The theoretical advantages and limitations of the DL implementation are related to the advantages and limitations of the vocabulary and graphical information model developed to address Hypothesis 1. Whereas, the implementation advantages and limitations are tied to the fundamentals of DL representations and software development.

An advantage of the DL representation is the formalization of the vocabulary for capturing the semantics of cDSPs and analysis support models. The DL representation is

an essential component in realizing a formal language for sharing and integrating decision-related information in a distributed design environment. The vocabulary developed in Hypothesis 1 enables disciplinary analysis models to be represented and subsequently integrated into multi-disciplinary design decisions. However, the key advantage of DL is the realization of the vocabulary in a computer-based representation. DL is used as the formalism to represent the semantics of the cDSP decision construct in a means that can be shared via extended computer-based networks. Thus, the DL representation addresses a major limitation of the graphical information model by providing a means through which the semantics of design decisions are computationally represented. The DL representation of the base vocabulary can be used in conjunction with DL constructs to develop complex concept definitions.

The fundamental theoretical limitations of the DL representation are tied directly back to the limitations of the vocabulary, modeling approach, and scope of applicability. It can be argued that the fundamental limitation of the DL representation is the scope of applicability of the formal language. For example, the DL representation provides a declarative approach for capturing the information associated with cDSP and analysis models. However, the representation is not tied to a solution technique of executable code. Thus additional effort is required to "translate" the declarative representation to an executable representation and/or to executable analysis models. Additionally, for existing simulation models, wrappers must be developed using the DL language to enable integration and information exchange. This requires that disciplinary experts and decision makers learn a new language for modeling decisions and analysis models. Another limitation of the DL representation is tied to the scope of applicability of the decision

vocabulary and DL representation. Currently, the vocabulary enables designers to model decision as cDSP and closely related concepts. Similarly, the DL representation is limited to declarative representation of equation-based and computationally-based simulation models. However, there are many different mathematical formulations of multi-objective design decisions that may be used and analysis models that may be used for design decisions.

A key implementation advantage of developing information models with the DL formalism are due to the mathematical foundations on which description logic is based. The mathematical properties of DL provide sound and consistent algorithms that can be exploited during the development and usage of information models. For example, DL support standard reasoning algorithms that are used to organize information hierarchically. As previously discussed, the scope of the vocabulary is limited to cDSPs and associated analysis models. However as presented in Chapter 4, DL provides support information extensibility and consistency and enables designers to incrementally specify and extend the scope of applicability while maintain information consistently. The extensibility of the vocabulary is illustrated in Section 4.5.4 through the representation of robust design decisions. A implementation limitation of the DL representation is the overhead required to capture and represent decision-related information. The proof-of-concept development environment utilized in this dissertation for modeling design decisions using DL is comprised of Protégé-OWL, RacerPro, and RacerPorter. The software tools are cumbersome to use and represent an additional cost associated with modeling multi-disciplinary design decisions. Thus, to increase the usability and reduce the overall cost of the DL representation, specialized software tools and frameworks must

be developed that enable designer and analysts to seamlessly capture decision-related information using the DL vocabulary. Finally, fundamental limitations exist in the DL tools. For example, while DL, in the richest form enables reasoning and organization to be performed on individuals the current generation of DL tools does not provide this capability. Additionally, novel reasoning algorithms such as *least common subsumer* have not been computationally implemented. The current limitations of DL software and tools limit the full capabilities that can be taken advantage of in the domain of engineering design. A detailed discussion about future software and tool development to realize the full potential of the DL representation is presented in Section 7.4.

7.2.3 Sub-Research Question 3 - Organization and Retrieval of Decision-Related Information

The third research question addresses the need to organize and check the information models for consistency. This question is motivated by the fact that the information associated with design decision is specified by several designers: "*How can decision-related knowledge be organized and retrieved to enable reuse?"* The hypothesis for the third research question addresses the need to dynamically organize and validate engineering information models. Hypothesis 3 is *"Reasoning and querying services supported by Description Logic can be utilized for organizing and retrieving decision related information."* Hypothesis 3 is motivated by the need to reuse engineering knowledge throughout the product realization process and across design decisions.

Theoretical structural validity is performed by examining the properties of several information modeling formalisms. As addressed in Hypothesis 2, three modeling formalisms are discussed in the context of several information modeling requirements

including maintaining consistency of the information base and enabling dynamic organization of the concepts specified in the information base. The theoretical underpinnings of DL and mathematical properties enable sound and complete reasoning to be performed on information models specified using DL. For example, It has been shown in literature that the value of DL formalisms for conceptual information modeling is using reasoning algorithms to maintain information model consistency, dynamically organize concepts in a hierarchical manner, and ensure concepts are accurately modeled. Thus, the theoretical structural validity of DL is accepted based on a critical review of relevant literature and more importantly on the mathematical properties and foundations of DL.

In the same manner as Hypothesis 1 and 2, empirical validation of Hypothesis 3 is completed by establishing the appropriateness of the examples and then testing the examples in the context of the hypothesis. The examples completed in this dissertation are appropriate for validating Hypothesis 3 because they enable the extensibility, dynamic organization, and consistency of the information to be maintained. As previously stated, the example are sufficient real and complex, thus support empirical structural validity. The empirical performance validity of Hypothesis 3 is completed by testing the information organization and consistency of the information base using DL reasoning algorithms. The core decision and analysis models presented in Chapter 4 are inherently related through subclass/superclass relationships. For example, a concept definition that captures the general information associated with an analysis model is first specified in the information base. After publishing the concept to the information model, the vocabulary is extended and two new specialized concepts of equation-based and

computational analysis models are created. However, the subclass relationships between the concept definitions are not explicitly and must be determined based on DL reasoning algorithms.

Additionally, the order in which the concept definitions are specified in the information are arbitrary (i.e., specific to general and general to specific). Specializations of cDSP and analysis model concepts are created for modeling the decisions associated with the cantilever beam and heat sink design problems. It is shown in Chapters 5 and 6, that DL reasoning algorithms provide a means for creating a hierarchy of design decisions and analysis model dynamically and independent of the order of creation. Additionally, as new concepts are added or existing concepts are modified the DL reasoning algorithms will reclassify the concepts and create a new hierarchy based on the concept definitions. The language has been shown to be consistent as extensions are added to the vocabulary. For example, the robust design decision were not considered during the original development of the vocabulary. Additionally, the empirical performance validity of Hypothesis 3 is further supported by implementing the DL concept specifications in Protégé-OWL and reasoning with the concept definitions using the RacerPro/RacerPorter DL reasoner. The results obtained from RacerPro/RacerPorter are evaluated from an incremental view. For example, the various states of the decision information model are verified to ensure the changes are consistent with the concept definitions. The resulting hierarchy from the DL reasoners is consistent with expected results. Thus, the example problems and implementation are sufficient for establishing the empirical validity of Hypothesis 3.

As discussed, a main advantage of the DL representation is that it enables information to be organized in a hierarchical fashion based on concept definitions, not on explicit relationships. The reasoning capabilities supported in DL enable designers address the shortcomings associated with information organization and retrieval to support the formulation of engineering design decisions is addressed by leveraging the DL concept specifications and reasoning algorithms. For example, the subsumption algorithm enable designers to systematically specify or retrieve information from the information based in a manner that replicates the design process (i.e., general to specific or conceptual to detailed). However, a major limitation of the DL-based organization and retrieval is related to the lack of software tools that enable design decision makers to seamlessly retrieve and organize decision concepts. As previously noted, the DL representation requires that the designer learn a new language through which information can be modeled and subsequently retrieved. A discussion on next generation information modeling tools to address these issues is presented in Section 7.4.

7.3 Theoretical Performance Validity of the Research Hypotheses

The validity of the research question and hypotheses are discussed independently in the previous sections. In this section, the previous validation arguments are synthesized and the general validity of the formal language is argued as a whole. In Chapter 1, an overarching research question and hypothesis are posed. The primary research question is: *How can information from multiple sources be (a) systematically captured and (b) formally represented in a computational means to facilitate integration in decision-centric design?* The hypothesis that is formulated to address the research question is:

Description Logic-based information models will provide a formal language for integrating multi-disciplinary decision knowledge.

In this section, the theoretical performance validity of the formal language is established. Theoretical performance validity is aimed at building confidence in the generality of the method and accepting that the method is useful beyond the example problems. The general applicability of the formal language is partially established by assessing the characteristics of the example problems presented in this dissertation. In Chapter 3, several characteristics of engineering design problems are developed for evaluating information modeling formalisms. These characteristics include:

- The information model should be extensible.
- The information model should enable consistency checking of concepts.
- The information model should provide information organization capabilities:

Specific requirements for modeling multi-objective engineering design decisions and analysis models included:

- The information model should enable designers to systematically capture and represent design problem knowledge from multiple perspectives.
- The information model should provide a computer-interpretable means for representing decision-related knowledge that can be exchanged and shared amongst stakeholders.
- The information model should support the integration of analysis models from multiple disciplines and unambiguous representation of analysis-related knowledge.

- The information model should support the reuse and retrieval of decision-related knowledge.
- The information model should have well-defined structure and pre-defined vocabulary of symbols to enable collaboration between decision makers.
- The information model should enable the limitations and assumptions of analysis models to be captured.
- The information model should be easy to understand by engineering designer.

The formal language developed in this dissertation satisfies these requirements (see Table 4-10). Additionally, several design examples are developed to validate the hypotheses posed in this research. Thus, this involves establishing that the example problems are representative of a general class of engineering design problems. The general characteristics of problems for arguing the validity of the formal language developed in this dissertation are summarized as:

- A systematic method can be developed for explicitly capturing and formulating engineering design problems as optimization problems. A logical set of phases and steps can be established to support the systematic formulation and integration of information from multiple domains.
- The decisions encountered in design can be formulated mathematically as compromise decision support problems (cDSPs). This is generally true when mathematical representations of the design requirements, system behaviors, and design representations exist.

- Design decisions require the integration of information from multiple domains through the linking and coupling of engineering analysis models. The complex behavior of engineering systems can be simulated through mathematically-based models. These models can be integrated to support design decisions.
- Disciplinary analysis models can be developed independent of a particular design problem and can be used in many different design scenarios. Disciplinary analysis models can be represented as equation-based or parameterized computational models. The information associated with engineering analysis models can be explicitly captured external to a particular design scenario or application
- Disciplinary analysis models are organized in a hierarchical fashion from the general to the specific. Disciplinary analysis models can be organized from the general to the specific based on properties of the information associated with the models. The models can be organized based on assumptions and quantities
- Disciplinary analysts and multi-disciplinary decision makers communicate through a predetermined set of quantities. In order to enable the integration and coupling of analysis models and design decision, the stakeholder in the design process must commit to an established vocabulary of design parameters.
- The complex information associated with design decisions can be represented using a predetermined set of vocabulary and grammar. The grammar for capturing the semantics and structure of design decisions. Information modeling concepts can be exploited to enable the representation and subsequently the exchange of information associated with engineering design decisions.

The formal language is shown to be useful for addressing the example problems in this dissertation. Each of the example problems demonstrated in this dissertation exhibit several of the characteristics. Hence, we are confident that the formal language can be used for a general class of engineering problems that satisfy the afore-mentioned characteristics.

The four quadrants of the validation square are addressed and the research hypotheses are validated and the limitations are discussed. Thus, the overarching research objective is achieved, namely: to *"develop a computational representation (i.e., a formal language) for capturing the semantics of engineering design decisions to facilitate the integration of information from multiple design perspectives."* We are confident that the formal language developed in this research is applicable for a general class of engineering problems because the language is extensible and robust. Having established the validity of the research hypotheses and answered the research question, the next step is to discuss the contributions from this research.

7.4 Contributions

The primary research contribution in this research corresponds to the overarching goal, primary research question, and primary research hypothesis – *a computational knowledge representation (i.e., a formal language) for capturing the semantics of engineering design decisions in a structured manner to facilitate the exchange and integration of knowledge from multiple design perspectives throughout the product realization process.* The primary contribution is expanded into several secondary contributions. The contributions in this research are in the fields of design method and computer-based information management for engineering design.

A high level contribution in this dissertation is **the identification of a need for advanced information representations and engineering information management for multi-disciplinary decision making.** This contribution is achieved through a critical review of existing literature and development in the area of multi-disciplinary design decision making and technologies and software for enabling the integration of multiple engineering disciplines and domains. The current state of the art in computer support for engineering design decisions and multi-disciplinary design optimization frameworks is critically reviewed. It was determined based on the review of current literature that decision support frameworks have not adequately addressed the need to developed information models and formalized languages for exchanging disciplinary information in the context of multi-disciplinary design decisions. While there has been a substantial effort in the development of standardized product models and information representations, these developments have not permeated decision making frameworks. The need for communication protocol and advanced information management, identified in the early 1990's, has not been adequately addressed, ultimately resulting in *ad-hoc* approaches for integrating disciplinary information for supporting design decision. Researchers have concentrated on the execution and coordination of design decisions, but have not developed information representations for capturing and reasoning with decision related knowledge. While the claim is made that information must be exchanged between disciplines, the gap has not been adequately addressed.

Thus, the gaps and limitations in current decision support frameworks leads to the second contribution in this research, the development of a standardized information model and formal language for capturing the semantics of engineering design decision. **A**

formal language is developed for capturing the semantics of engineering design decisions modeled as cDSP and associated engineering analysis models. The formal language consists of four closely related components including: (1) a systematic method for formulating engineering design decision, (2) a vocabulary of the concepts and properties associated with multi-objective design decisions that constitute the formal language, (3) a graphical representation and notation of the information models for multi-objective design decisions and analysis models, and (4) DL concept definitions based on the vocabulary that provide a computer interpretable representation. The components, collectively, provide a means for unambiguously representing and exchanging decision-related information between multiple engineering design disciplines. **The systematic method provides a structured means for explicitly capturing the information associated with engineering design decisions.** The systematic method is based on current literature for modeling multi-objective design decisions as compromise decision support problems. The cDSP is a well-accepted formulation for modeling decision commonly encountered in engineering design. The method provides a basis for capturing and integrating decision-related information from multiple perspectives and consists of seven phases. Mathematical-based representations of engineering decision information are developed based on the phases and steps for decision formulation. These mathematical representations are then translated into traditional information modeling representations. The mathematical representations are used for developing the vocabulary, graphical information models, and DL implementation.

The method is decomposed into two closely-related sub-methodologies. The first sub-method is focused on capturing the "core" information associated with engineering

decisions including the representation of design variables, design parameters, system goals, design and analysis constraints, and decision preferences. The first sub-method consists of five phases and provides a means for structuring the information associated with a design decision without committing to specific analysis model.

The second sub-method comprises a single phase consisting of four steps. The second sub-method provide as structured basis for explicitly capturing analysis-related information. The second sub-method is targeted towards disciplinary analysis experts by providing a mean for publishing information about complex engineering analysis models that enable seamless integration with engineering design decisions. This decomposition is chosen to enable disciplinary analysts to systematically capture and "publish" analysis models independent of design decisions.

The first sub-method encapsulates the phases and steps that are associated with multi-disciplinary decision making. The information captured in the first method provides a means for declaring the information of interest for a design decision. However, sufficient freedom remains that enables disciplinary experts from multiple domains to develop and implement different analysis models. Conversely, the interface between the first and second sub-method provides a means for developing analysis models somewhat independently of the decisions in which they may be used. The advantages of these decompositions lie in the fact that the multi-disciplinary decision formulation and the implementation of disciplinary analysis models are decoupled. This implies that not all the information associated with a complex multi-disciplinary design decision must be exchanged between all disciplines. Only those disciplines that must share or exchange information must communicate, thus simplifying and reducing the

shared information between disciplinary analysis models while retaining the complex inter-disciplinary information exchange at the decision level.

A vocabulary is developed based on the mathematical information representations identified from the systematic method. **The vocabulary consists of a predetermined set of concepts and properties used for describing the information associated with multi-objective engineering design decisions**. The semantics of the vocabulary are established by developing definitions of each of the basic concepts and properties. In addition, the domain and range of properties are specified, thus restricting the properties to a predetermined set of concepts. The vocabulary provides several advantages, some of which are indirectly discussed in the systematic method section, for modeling the information associated with engineering decisions. First, the vocabulary is formal. In this context, formal refers to explicitly capturing and representing the entities and relationship for a domain of discourse in a formal language. In this research, the formal language for representing engineering decisions is based on a description logic language and the proposed vocabulary. The vocabulary is developed such that disciplinary analyst are able to capture the quantities and constraint associated with analysis models, without knowing how the analysis models will be coupled to other discipline and independent of specific design decisions. Additionally, the vocabulary enables decision makers to model complex engineering decisions using a basic set of concepts and properties.

The graphical information models provide a means for visualizing the information and relationships associated with engineering design decisions. The representations become increasingly complex and the number of relationships becomes denser as does the complexity of the decision concepts. The graphical representation

enable the "linkages" between the quantities (i.e., design variables, parameters, and system goals) to be visualized and the relationships between these parameters and design constraints and analysis models. Finally, the graphical representation serves as a prescriptive template of what information must be instantiated. Additionally, the graphical representation enables a designer to determine what analysis models are driven a design decision and how the analysis models are coupled.

The final component of the formal language is the DL implementations of the decision-related concepts. **DL provides a means for modeling the semantics of cDSPs and analysis models in a computationally processible representation**. DL uses a fixed set of primitives (i.e., concepts and properties) can be used to construct complex object descriptions. DL is used in conjunction with the base vocabulary for developing formal definitions of decision information and analysis models. Several information representations are developed (e.g., traditional cDSP, Robust I cDSP, Robust II cDSP, Robust I-II cDSP, and analysis model concept) using the vocabulary and DL constructs. Complex concepts are defined using a predetermined set of construct and vocabulary. As illustrated the DL representation provide a computer-interpretable representation of decision related knowledge that can be reasoned with.

Collectively, the systematic method, vocabulary, graphical representation, and DL implementation provide a means for enabling designers to explicitly capture the information associated with multi-objective design decisions in a manner that is human-interpretable and computer-processible. Most importantly, the language and implementation developed in this research provide a digital interface for integrating and exchanging decision-related information in multi-disciplinary design problem.

The development of digital interfaces for integrating and exchanging information in the context of engineering design decisions has been the focus of several research efforts. However, previous research efforts have not adequately addressed the need for formal information representations to enable the development of digital interfaces. In this research, we approach **the development of digital interfaces from a computational perspective by establishing a formal language for representing engineering design decisions**. We believe that this approach enables us to focus on the *digital* aspect and create representations that enable the exchange of product information. The underlying notion in this research is that *formal information model and vocabulary will provide a computational digital interface for exchanging information associated with engineering design decisions.* Hence, digital interfaces are emergent concepts that represent formal representation of decision-related information. With this approach were are able to address the philosophical level digital interface, by creating representation of design decisions that enable disciplinary information to be packaged and exchanged.

The final contribution in this research is **a critical analysis and application of data modeling and knowledge representational technologies in engineering design.** Engineering information management (EIM), specifically the development of information models, is becoming increasingly important to facilitating the exchange of digital product information across the extended enterprise. A myriad of information models have been proposed for capturing a broad scope of design information. Recently, description logics (DLs) have received significant attention in current literature as the underlying representational formalism for developing engineering information models. In this paper, we address the question: *"Why should description logics (DLs) be used for*

engineering information management (EIM)?" We identify the characteristics of engineering design problems and the requirements for EIM, review common information modeling formalisms, and critically evaluate the benefits of DLs over other representational formalisms. The use of DLs is illustrated for modeling engineering decision knowledge. DLs offer advantages over other formalisms by supporting a logics-based representation and a set of reasoning algorithms.

7.5 Opportunities for Future Work

The research completed in this dissertation is focused on developing a formal language for enabling the integration of information to support cDSP formulation. In addressing the research questions and hypothesis we have identified limitations of the research. These limitations should not be thought of as endpoint, but rather that they provide several opportunities for continued research and development. In this section, we present an overview of the future work to enhance the research contributions and address the limitations.

Increased integration with existing design support tools. The overall goal of this research is to develop a representation that enables 1) the semantics of engineering design decisions and analysis models to be captured and 2) to provide a mean to integrate and exchange information to enable the formulation of engineering design decisions. While the formal language has been developed to capture the semantics of design decision, the integration of information generated throughout the design process with existing tools has not fully been addressed. Thus, an avenue of additional research is establishing interfaces between commonly used design support tools such as CAD, FEA, CFD, and materials databases and decision support frameworks. This will enable engineering designer to

leverage existing information generation tools, and facilitate integration of the product information.

Seamless creation of executable representations. The formal language developed provides a tool for capturing the semantics of design decisions. However, the formal language provides a declarative representation of decision related knowledge and is not tied to any particular solution approach. Thus, significant manual effort is required to created executable representations of the declarative decision information. This manual linking requires significant cost, can be affected by human error, and the consistency between the declarative and executable representation must be maintained. Thus, another avenue of further research is the development of techniques for capturing the relationships between the two representations. This will enable designers to concentrate on the declarative representations of the decision-related information, while addressing the need for executable representations.

Extension of the base vocabulary. The vocabulary established in this research is developed for a targeted scope of engineering design decisions, namely the cDSP and supporting analysis models. However as illustrated in Chapter 4, there are several variations of the cDSP construct and many other different formulations of design decisions. Thus, research associated with the development of additional terminology for modeling other types of engineering design decision is needed. For example, the language must be extended to handle many different types of engineering analysis model such as empirical data, tables, and databases. This will significantly extend the scope the language, but provide a more robust and valuable language. A process similar to that

presented in Chapter 4 must be completed to explicitly capture the information structure associated with the inclusion of additional design decisions.

Development of computer-based framework. A common limitation discussed throughout this research is the lack of a computer-based framework for capturing and using decision information. Currently, the formal language is not transparent to the designer. In other words, significant effort is required to represent the information associated with design decisions and analysis models. Decision makers must learn a new language for explicitly modeling the information associated with design decision. While the formal language does provide significant value, the perceived value may be much less due to the overhead required to use the representation. Additionally, the intent in this research is not to create a end user application, but rather lay the foundation for new developments based on the language. Thus, a major avenue for future research and development is the specification of architecture and collaborative support tools based on the formal language.

7.6 Closing Thoughts

In completing this research I was astonished by the gap between the domains of decision support frameworks and engineering information management. There are a myriad of research opportunities in each of these domains that must be addressed individually. However, there is tremendous potential in the synergistic development of standardized information representations and formal languages for enabling effective communication for engineering decision support. In particular, information models are needed to capture the semantics of engineering design decisions, thus facilitating

information exchange and integration across extended networks between design disciplines.

In this thesis, I believe I have taken the first few steps and have begun to lay the groundwork for leveraging information modeling approaches to enable effective communication and information exchange in the context of multi-disciplinary decision-based design. The formal language developed in this research is a small, but integral part of collaborative product development and product life-cycle management (PLM). I believe the formal language will enable engineering designers to communicate across extended networks and capture the interdisciplinary nature of addressing complex engineering design problems. My goal in this research is not meant to replace or invalidate current information modeling efforts, such as STEP, or decision support framework. Rather, the goal in developing the formal language is to provide a higher level of abstraction and enhance the capabilities of current approaches by expanding the scope of information captured throughout the design process. For example, current information models provide a means for representing and exchanging detailed geometric representation and analysis data, but also to communicate the decision in which the geometric data and analysis models are used. In establishing the formal language, we believe that we are a step closer to being able to model engineering design as a decision-centric process. However as noted throughout this dissertation, the primary limitations of the formal language are due to implementation-level deficiencies. For example, I believe designers and disciplinary analysts should be able to explicitly capture decision-related information using the formal language in a seamless manner, integrate disciplinary analysis models into design decision, link existing design representations such as CAD

geometry and materials databases, and the declarative information representations should be related to executable decision representations. However, addressing the implementation-level deficiencies go beyond simply *creating a big computer program.* It is envisioned that additional research is needed to address the architecture and approach for integrating existing design support tools and development and refinement of product information models to enable integration with design decisions. I believe *that systems integration through information* addressed in this dissertation is a powerful approach for modeling the design process and that information modeling formalism provide a means by which multiple engineering disciplines can be seamlessly integrated. Finally, I believe the research completed and the steps taken in this dissertation provide a foundation on which next-generation product development tools and approaches can be developed and design research can be conducted.

APPENDIX A:

CANTILEVER BEAM MATLAB CODE

The following MATLAB code is used to modeling the cantilever beam design problem and analysis models. The code includes functions that represent the analysis models and the compromise decision support problem.

```
%Author: Gregory Mocko
%Date: July 2005
%This program is the cDSP decision model for the cantilever beam end load.
%------------------------------------------------------------------------

%
binitial = 0.01;% Beam width
hinitial = 0.01;% Beam height
blower = 0.01;% beam width
hlower = 0.01;% beam height
bupper = 0.1;% beam width
hupper = 0.1;% beam height

%Design Parameters
P=150; % Force applied at end of beam - 160 N
L=0.5; % Length of beam  - 0.5 meters
E=205; % Modulus of elasticity 200 GPa
G = 80e9; %Shear modulus 80 Pa
rho = 7872; %Density of steel kg/m^3
SY = 340; % Ultimate tensile strength 340MPa

targetweight_matrix = [3];
targetdeflection_matrix = [3e-6];
ResultsIndex = 1;

for twindex = 1:1:length(targetweight_matrix);
for tdindex = 1:1:length(targetdeflection_matrix);
targetweight = targetweight_matrix(twindex);
targetdeflection = targetdeflection_matrix(tdindex);

wweight = -0.1;

deflection_index=1;
for m = 1:1:11
wweight = wweight+0.1;
wdeflection = 1.0-wweight;
if wweight >= -0.05 & wweight <= 0.05
wweight = 0.0;
wdeflection = 1.0;
end
if wweight >= 0.95 & wweight <=1.05
wweight = 1.0;
wdeflection = 0.0;
end
```

```
b = 0.01; % beam width
h = 0.01; % beam height
bsteps = 20;
hsteps = 20;
bincrement = (bupper-blower)/(bsteps-1);
hincrement = (hupper-hlower)/(hsteps-1);
%wweight = 0.5;
%wdeflection = 0.5;
bstart = blower;
hstart = hlower;
b = bstart;
h = hstart;

counter = 0;
for i = 1:1:bsteps
for j = 1:1:hsteps
counter = counter+1;

%Call the appropriate models
[deflection,beam_angle] = beam_deflection_model(L,b,h,P,E);
weight = beam_weight(L,b,h,rho);
stress = beam_stress_model(L,b,h,P);
Fcritical = beam_buckling_model(b,h,P,L,E,G);

%--------------------------------------------------------------------------

% Calculation of Objective Function
devdeflection = 1 - (targetdeflection_matrix(tdindex)/deflection);
devweight = 1 - (targetweight_matrix(twindex)/weight);
Objectivefunction = wdeflection*devdeflection + wweight*devweight;
%--------------------------------------------------------------------------
% Calculation of Constraints
if L >= 10*b
Slender_constraint = 1;
else
Slender_constraint = 0;
end

if L >= 10*h
Long_constraint = 1;
else
Long_constraint = 0;
end

if (SY*10^6 > stress)
elastic_deformation_constraint = 1;
else
elastic_deformation_constraint = 0;
end

if (Fcritical > stress)
buckling_constraint = 1;
else
buckling_constraint = 0;
end

acceptable_percent_error = 0.05;
error1 = beam_angle - sin(beam_angle);
error = abs(error1/sin(beam_angle));
if error <= acceptable_percent_error
small_deflection_constraint = 1;
else
small_deflection_constraint = 0;
```

```
end

%-------------------------------------------------------------------------
%ResultsIndex = hsteps*(i-1)+j;
% Results matrix
Results(1,ResultsIndex) = h;
Results(2,ResultsIndex) = b;
Results(3,ResultsIndex) = Objectivefunction;
Results(4,ResultsIndex) = weight;
Results(5,ResultsIndex) = deflection;
Results(6,ResultsIndex) = stress;
Results(7,ResultsIndex) = Slender_constraint;
Results(8,ResultsIndex) = Long_constraint;
Results(9,ResultsIndex) = elastic_deformation_constraint;
Results(10,ResultsIndex) = buckling_constraint;
Results(11,ResultsIndex) = Fcritical;
Results(12,ResultsIndex) = small_deflection_constraint;
Results(13,ResultsIndex) =
Results(7,ResultsIndex)*Results(8,ResultsIndex)*Results(9,ResultsIndex)*Result
s(10,ResultsIndex)*Results(12,ResultsIndex);
Results(14,ResultsIndex) = wweight;
Results(15,ResultsIndex) = wdeflection;
Results(16,ResultsIndex) = targetweight_matrix(twindex);
Results(17,ResultsIndex) = targetdeflection_matrix(tdindex);
Results(18,ResultsIndex) = beam_angle;
%-------------------------------------------------------------------------
%Create a square martix for plotting purposes
h_matrix (i,j) = h;
b_matrix (i,j) = b;
objectivefunction_matrix(i,j) = Objectivefunction;
solution(counter,1) = h;
solution(counter,2) = b;
solution(counter,3)=  Objectivefunction;
solution(counter,4) =
Slender_constraint*Long_constraint*elastic_deformation_constraint*buckling_con
straint*small_deflection_constraint;

[valid_solution_i] = find(solution(:,4) == 1);
valid_solution=solution(valid_solution_i,:);
%[min_objective_function = min(objectivefunction_matrix);
%min_valid_soluti
%-------------------------------------------------------------------------
ResultsIndex = ResultsIndex + 1;
h = h + hincrement;
end %increment b
h = hstart;
b = b+bincrement;
end %increment h
[solution_obj, solution_ind] = min(solution(:,3));
decision_solution(m,:) = solution(solution_ind,:);
[valid_solution_obj, valid_solution_ind] = min(valid_solution(:,3));
valid_decision_solution(m,:) = valid_solution(valid_solution_ind,:);
%pause(0.5);
%-------------------------------------------------------------------------
 [i,j]=find(Results(13,:)==1);
h_v = Results(1,j);
b_v = Results(2,j);
obj_v = Results(3,j);
K = convhull(h_v,b_v);
%-------------------------------------------------------------------------
figure1 = figure;
axes1 = axes('Parent',figure1);
hold on
```

```
%hold(axes1,'all');
xlabel('h (m)');
ylabel('b (m)');
xlim([hlower hupper]);
ylim([blower bupper]);
plot_title = ['Design space for beam (w_w_e_i_g_h_t =',num2str(wweight),'
w_d_e_f_l_e_c_t_i_o_n = ',num2str(wdeflection),')'];
title(plot_title);
[C,h] = contour(h_matrix,b_matrix,objectivefunction_matrix);
clabel(C,h)
%axis ([hlower hupper hupper bupper])
%colorbar1 = colorbar;
plot(h_v(K),b_v(K),'k-','LineWidth',2);
plot(decision_solution(m,1),decision_solution(m,2),'Color',[1 0
0],'LineStyle','none','Marker','square','MarkerSize',10,'MarkerEdgeColor',[1 0
0],'MarkerFaceColor',[1 0 0]);
plot(valid_decision_solution(m,1),valid_decision_solution(m,2),'Color',[0 0
1],'LineStyle','none','Marker','o','MarkerSize',10,'MarkerEdgeColor',[0 0.498
0],'MarkerFaceColor',[0 0.498 0]);

% Plot the lines on top - line plots
%--------------------------------------------------------------------------
figure2 = figure;
axes2 = axes('CameraPosition',[-0.1332 -0.6733 1.661],...
'CameraUpVector',[0.1909 0.7616 2.534],...
'Parent',figure2);
%hold(axes2,'all');
hold on
xlabel('h (m)');
ylabel('b (m)');
zlabel('Objective Function');
title(plot_title);
xlim([hlower hupper]);
ylim([blower bupper]);

surf1 = surf(h_matrix,b_matrix,objectivefunction_matrix);
movie_frame(m) = getframe;

end %increment objective weights
end %wdindex
end %wwindex

%--------------------------------------------------------------------------
figure3 = figure;
% Create axes
axes3 = axes('Parent',figure3);
%hold(axes3,'all');
xlabel(axes3,'h (m)');
ylabel(axes3,'b (m)');
hold(axes3,'all')
title(axes3, 'Plot of design space for beam');
[valid_i] = find(Results(13,:)==1);
valid_h = Results(1,valid_i);
valid_b = Results(2,valid_i);
scatter(valid_h,valid_b,'or');

[invalid_i] = find(Results(13,:)~=1);
invalid_h = Results(1,invalid_i);
invalid_b = Results(2,invalid_i);
scatter(invalid_h,invalid_b,'+b');
legend1 = legend(axes3,'Valid Design Space','Invalid Design Space');

%--------------------------------------------------------------------------
```

```
function [weight] = beam_weight(L,b,h,rho)
%Author: Gregory Mocko
%Date: July 2005
%This function is used to weight of the beam
%
%input
%   L - beam length (m)
%   b - beam width (m)
%   h - beam height (m)
%   rho - density of hte beam (kg/m^3)
%   g - gravity (9.81 m/sec^2)

%output
%   weight - Beam weight (N)
%--------------------------------------------------------------------------
g = 9.81; %m/s^2
weight = h*b*L*rho*g;
end

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

function [stress] = beam_stress_model(L,b,h,P)
%Author: Gregory Mocko
%Date: July 2005
%This function is used to maximum normal stress in the beam
%
%input
%   L - beam length (m)
%   b - beam width (m)
%   h - beam height (m)
%   P - Load (kN)

%output
%   stress - Maximum normal stress in beam (MPa)
%--------------------------------------------------------------------------
stress = P*L*(h/2)/((1/12)*b*h^3);
end

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

function [Fcritical] = beam_buckling_model(b,h,P,L,E,G)
%Author: Gregory Mocko
%Date: July 2005
%This function is used to determine lateral torsional buckling of beams in
bending
%
%input
%   b - beam width (m)
%   h - beam height (m)
%   P - Load (kN)
%   L - beam length (m)
%   E - Modulus of elasticity (GPa)
%   G - Shear modulus (GPa)
%
%output
%   Fcritical - Critical stress to cause buckling (MPa)
%--------------------------------------------------------------------------
%Maximum Stress
I_min = (1/12)*b^3*h;
I_max = (1/12)*b*h^3;
J = (1/12)*b*h*(b^2+h^2); %polar moment of inertia of rectangular beam
```

```
L_effective = 0.783*L/(1-2*h/L);
alpha = sqrt(2*pi)*((E*10^9*I_min)/(G*J))^0.25*(sqrt(L_effective*h)/b);
Fcritical = (pi^2*E*10^9)/alpha^2;



%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

function [deflection,angle] = beam_deflection_model(L,b,h,P,E)
%Author: Gregory Mocko
%Date: July 2005
%This function is used to deflection and angle of the beam
%
%input
%   L - beam length (m)
%   b - beam width (m)
%   h - beam height (m)
%   P - Load (kN)
%   E - Modulus of elasticity (GPa)
%
%output
%   deflection - deflection at free end of beam (meters)
%   angle - angle at fixed end of beam (radians)
%-------------------------------------------------------------------------
deflection = (P*L^3)/(3*E*10^9*((1/12)*b*h^3));
angle = (P*L^2)/(2*E*10^9*((1/12)*b*h^3));

end
```

APPENDIX B:

HEAT SINK MATLAB CODE

The following MATLAB code is used to modeling the fin array heat sink design problem and analysis models. The code includes functions that represent the analysis models and the compromise decision support problem.

```
%% Version information
%-----------------------------------------------------------------
%Author: Gregory Mocko
%Date: September 2005
%This function completes an exhaustive search for the heat sink cDSP
%-----------------------------------------------------------------------

%% Heat Sink Design Parameters
% Material properties of fluid
% Properties of Air
k_air = 0.026; %W/m-k
rho_air = 1.161; %kg/m^3
k_aluminum = 237; %W/m-k
E_aluminum = 69e9;%69e9Pa
aluminum_sy = 130e6;%130e6Pa
rho_aluminum = 2702; %2702kg/m3
mu_air = 208.2e-7; %N-s/m^2
min_stiffness = 5250000; %N/m
volumetric_flow_rate = 0.0047; %m^3/sec - 10CFM

Heat_sink_load = 300;%N
chip_power = 50; %W
ambient_temperature = 318; %K
max_chip_temperature = 343; %K
heat_sink_width = 0.08; %meters
heat_sink_length = 0.08; %meters
heat_sink_base_thickness = 0.006; %meters

%% Heat Sink Design Variables
% Design Variables
%x(1) = fin_length
%x(2) = fin_width
%x(3) = Number_of_fins

%% Heat Sink cDSP Parameters
target_resistance = .10;%K/watts
target_space = 0.00001;%m^3
target_weight = .010; %kg

%% Weighting Value
weight_resistance = 0.33;
weight_space = 0.33;
weight_weight = 0.33 ;
```

```
%% Specification of lower and upper limits on design variables
fin_length_lower = 0.01;%
fin_width_lower = 0.0001;%
Number_of_fins_lower = 28;%

fin_length_upper = 0.05;%
fin_width_upper = 0.0005;%
Number_of_fins_upper = 30;%

%% Step size for exhaustive searching
fin_length_steps = 10;
fin_width_steps = 10;
Number_of_fins_steps = Number_of_fins_upper - Number_of_fins_lower + 1;

%% Initialization of Results matrix for computational efficiency
Results =zeros(fin_length_steps*fin_width_steps*Number_of_fins_steps,22);

fin_length_increment = (fin_length_upper-fin_length_lower)/(fin_length_steps-
1);
fin_width_increment = (fin_width_upper-fin_width_lower)/(fin_width_steps-1);
Number_of_fins_incrememt = 1;

fin_length_start = fin_length_lower;%
fin_width_start = fin_width_lower;%
Number_of_fins_start = Number_of_fins_lower;%

fin_length = fin_length_start;%
fin_width = fin_width_start;%
Number_of_fins = Number_of_fins_start;%

%% Initialization of for loops
ResultsIndex = 0;
for k = 1:1:Number_of_fins_steps
counter = 0;
for i = 1:1:fin_length_steps
for j = 1:1:fin_width_steps
counter = counter + 1;
ResultsIndex = ResultsIndex+ 1;
channel_width = (heat_sink_width - Number_of_fins*fin_width) / (Number_of_fins
- 1);

%Call the analysis models
%=====================================================
[resistance,h_bar] = Heat_Sink_Resistance(fin_length,fin_width,Number_of_fins,
heat_sink_length,heat_sink_width,k_air,k_aluminum,heat_sink_base_thickness,cha
nnel_width);
space = Heat_Sink_Volume(fin_length,heat_sink_length, heat_sink_width,
heat_sink_base_thickness);
weight = Heat_Sink_Weight(rho_aluminum, fin_length,fin_width,
heat_sink_length, heat_sink_width, heat_sink_base_thickness,Number_of_fins);
stiffness = Heat_Sink_Stiffness(E_aluminum, fin_length,fin_width,
heat_sink_length, heat_sink_width, heat_sink_base_thickness,Number_of_fins);
p_critical = Heat_Sink_Buckling(E_aluminum, fin_length,fin_width,
heat_sink_length);
[fin_stress,fin_load] =
Heat_Sink_Stress(fin_width,heat_sink_length,Number_of_fins, Heat_sink_load);
chip_temp = Heat_Sink_Chip_Temp(chip_power,resistance,ambient_temperature);
h_bar_matrix(ResultsIndex,1) = h_bar;
h_fully(j,i) = h_bar;

% Calculation of Objective Function
%=====================================================
dev_resistance = 1 - (target_resistance/resistance);
```

```
dev_space = 1 - (target_space/space);
dev_weight = 1 - (target_weight/weight);
Objectivefunction = weight_resistance*dev_resistance + weight_space*dev_space
+ weight_weight*dev_weight;

%=====================================================
% Calculation of Constraints
if fin_width * Number_of_fins < heat_sink_width
width_geometric_constraint = 1;
else
width_geometric_constraint = 0;
end

if channel_width > 0
channel_width_constraint = 1;
else
channel_width_constraint = 0;
end

velocity_air = volumetric_flow_rate /
(channel_width*fin_length*(Number_of_fins-1));
v(ResultsIndex) = velocity_air;
Area_channel = channel_width * fin_length;
perimeter_channel = channel_width + 2*fin_length;
Dh = 4*Area_channel / perimeter_channel;
Re_number = rho_air * velocity_air * Dh / mu_air;
% Check constraints and assumptions
%=====================================================

if Re_number <=2300
laminar_assumption = 1;
else
laminar_assumption = 0;
end

if fin_load <=p_critical
buckling_constraint = 1;
else
buckling_constraint = 0;
end

if (fin_stress <= aluminum_sy*10^6)
elastic_stress_constraint = 1;
else
elastic_stress_constraint = 0;
end
if (stiffness >= min_stiffness)
stiffness_constraint = 1;
else
stiffness_constraint = 0;
end

SMOI =
min(((1/12)*heat_sink_length*fin_width^3),((1/12)*heat_sink_length^3*fin_width
));
area= heat_sink_length*fin_width;
radius_gyration = sqrt(SMOI/area);
Leff = 2*fin_length;
slenderness = Leff/radius_gyration;

if slenderness > 66
elastic_buckling = 1;
else
```

```
elastic_buckling = 0;
end

c_t(ResultsIndex) = chip_temp;
if (chip_temp <= max_chip_temperature)
temperature_constraint = 1;
else
temperature_constraint = 0;
end

total_validity = width_geometric_constraint*channel_width_constraint*...
laminar_assumption*buckling_constraint*...
elastic_stress_constraint*stiffness_constraint*temperature_constraint;
design_validity = width_geometric_constraint*channel_width_constraint*...
stiffness_constraint*temperature_constraint;
analysis_validity = laminar_assumption*buckling_constraint*...
elastic_stress_constraint;

%=====================================================

% Results matrix - write results for all iterations
Results(ResultsIndex,1) = fin_length;
Results(ResultsIndex,2) = fin_width;
Results(ResultsIndex,3) = Number_of_fins;
Results(ResultsIndex,4) = Objectivefunction;
Results(ResultsIndex,5) = resistance;
Results(ResultsIndex,6) = space;
Results(ResultsIndex,7) = weight;
Results(ResultsIndex,8) = stiffness;
Results(ResultsIndex,9) = p_critical;
Results(ResultsIndex,10) = dev_resistance;
Results(ResultsIndex,11) = dev_space;
Results(ResultsIndex,12) = dev_weight;
Results(ResultsIndex,13) = width_geometric_constraint;
Results(ResultsIndex,14) = channel_width_constraint;
Results(ResultsIndex,15) = laminar_assumption;
Results(ResultsIndex,16) = buckling_constraint;
Results(ResultsIndex,17) = elastic_stress_constraint;
Results(ResultsIndex,18) = temperature_constraint;
Results(ResultsIndex,19) = stiffness_constraint;
Results(ResultsIndex,20) = total_validity;
Results(ResultsIndex,21) = design_validity;
Results(ResultsIndex,22) = analysis_validity;

% Results for each iteration on decision weighting preference
weight_results(counter,1) = fin_length;
weight_results(counter,2) = fin_width;
weight_results(counter,3) = Number_of_fins;
weight_results(counter,4) = Objectivefunction;
weight_results(counter,5) = resistance;
weight_results(counter,6) = space;
weight_results(counter,7) = weight;
weight_results(counter,8) = stiffness;
weight_results(counter,9) = p_critical;
weight_results(counter,10) = dev_resistance;
weight_results(counter,11) = dev_space;
weight_results(counter,12) = dev_weight;
weight_results(counter,13) = width_geometric_constraint;
weight_results(counter,14) = channel_width_constraint;
weight_results(counter,15) = laminar_assumption;
weight_results(counter,16) = buckling_constraint;
weight_results(counter,17) = elastic_stress_constraint;
weight_results(counter,18) = temperature_constraint;
```

```
weight_results(counter,19) = stiffness_constraint;
weight_results(counter,20) = total_validity;
weight_results(counter,21) = design_validity;
weight_results(counter,22) = analysis_validity;

% Write matrix for graphing purposes
%=====================================================

Objective_function_Matrix(i,j) = Objectivefunction;
fin_length_matrix(j) = fin_length;
fin_length = fin_length + fin_length_increment;
end % increment fin length
fin_width_matrix(i) = fin_width;
fin_length = fin_length_start;
fin_width = fin_width + fin_width_increment;
end % increment fin width

fin_width_fully (j,1) = fin_width;
fin_length_fully(i,1) = fin_length;

figure6 = figure;
%Create axes
axes6 = axes('Parent',figure6);
xlabel(axes6,'Fin Length (m)');
ylabel(axes6,'Fin Thickness (m)');
zlabel(axes6,'Number of Fins');
plot_title = ['h-bar value (Number of fins',num2str(Number_of_fins),')'];
title(axes6, plot_title);
view(axes6,[-37.5 30]);
%grid(axes6,'on');
hold(axes6,'all');
surf(fin_width_matrix,fin_length_matrix,h_fully)
%surf(fin_width_fully,fin_length_fully,h_fully)

%% Design Valid results -these results are not valid in accorance with design
%constraints but are for analysis constraints
[i,j]=find(weight_results(:,21)==1 & weight_results(:,22)==1);
dvalid_fin_length20 = weight_results(i,1);
dvalid_fin_thickness20 = weight_results(i,2);
dvalid_obj20 = weight_results(i,4);

[i,j]=find(weight_results(:,21)==0 & weight_results(:,22) == 1);
dinvalid_fin_length20 = weight_results(i,1);
dinvalid_fin_thickness20 = weight_results(i,2);
dinvalid_obj20 = weight_results(i,4);

% These results are valid for analysis constraints but not design
% constraints
[i,j]=find(weight_results(:,21) == 0 & weight_results(:,22)==0);
avalid_fin_length20 = weight_results(i,1);
avalid_fin_thickness20 = weight_results(i,2);
avalid_obj20 = weight_results(i,4);

%These results are not valid for analysis constraints
[i,j]=find(weight_results(:,21)==1 & weight_results(:,22)==0);
ainvalid_fin_length20 = weight_results(i,1);
ainvalid_fin_thickness20 = weight_results(i,2);
ainvalid_obj20 = weight_results(i,4);

K = convhull(dvalid_fin_length20,dvalid_fin_thickness20);

%% Find solution to cDSP for each value of Number of fins
[solution_objNF, solution_indNF] = min(weight_results(:,4));
```

```
decision_solutionNF = weight_results(solution_indNF,:);
[valid_i,valid_j]=find(weight_results(:,20)==1);
valid_solutionsNF = weight_results(valid_i,:);
[valid_solution_objNF, valid_solution_indNF] = min(valid_solutionsNF(:,4));
valid_decision_solutionNF = valid_solutionsNF(valid_solution_indNF,:);


%% Create figure
figure2 = figure;
axes2 = axes('Parent',figure2);
xlim(axes2,[fin_length_lower fin_length_upper]);
ylim(axes2,[fin_width_lower fin_width_upper]);
xlabel(axes2,'Fin Length (m)');
ylabel(axes2,'Fin Thickness (m)');
plot_title = ['Objective function values (Number of fins
=',num2str(Number_of_fins),')'];
title(plot_title);
hold(axes2,'all');
scatter(dvalid_fin_length20,dvalid_fin_thickness20,50,'or');
scatter(dinvalid_fin_length20,dinvalid_fin_thickness20,50,'+b');
scatter(avalid_fin_length20,avalid_fin_thickness20,50,'*g');
scatter(ainvalid_fin_length20,ainvalid_fin_thickness20,50,'xm');
legend4 = legend(axes2, 'Valid Design and Analysis ','Invalid Design, Valid
Analysis','Invalid Design and Analysis','Valid Design-Invalid
Analysis','Location','SouthOutside');
%% Here is the work
[C,h] = contour(fin_length_matrix,fin_width_matrix,Objective_function_Matrix);
clabel(C,h)
plot(dvalid_fin_length20(K),dvalid_fin_thickness20(K),'k-','LineWidth',2);
plot(decision_solutionNF(1,1),decision_solutionNF(1,2),'Color',[1 0
0],'LineStyle','none','Marker','square','MarkerSize',10,'MarkerEdgeColor',[1 0
0],'MarkerFaceColor',[1 0 0]);
if isempty(valid_decision_solutionNF)==0
plot(valid_decision_solutionNF(1,1),valid_decision_solutionNF(1,2),'Color',[0
0 1],'LineStyle','none','Marker','o','MarkerSize',10,'MarkerEdgeColor',[0
0.498 0],'MarkerFaceColor',[0 0.498 0]);
end
%
Number_of_fins = Number_of_fins + 1;
fin_width = fin_width_start;
end %increment number_of_fins
%==================================
%% Find Valid and Invalid Space
%Valid results
[i,j]=find(Results(:,20)==1);
valid_fin_length = Results(i,1);
valid_fin_thickness = Results(i,2);
valid_number_of_fins = Results(i,3);
valid_obj = Results(i,4);
valid_solutions = Results(i,:);

%Invalid results
[i,j]=find(Results(:,20)==0);
invalid_fin_length = Results(i,1);
invalid_fin_thickness = Results(i,2);
invalid_number_of_fins = Results(i,3);
invalid_obj = Results(i,4);

%% Find solution to cDSP
[solution_obj, solution_ind] = min(Results(:,4));
decision_solution = Results(solution_ind,:);
[valid_solution_obj, valid_solution_ind] = min(valid_solutions(:,4));
valid_decision_solution = valid_solutions(valid_solution_ind,:);
```

```
%% Create figure of valid and invalid space
figure1 = figure;
% Create axes
axes1 = axes('Parent',figure1);
xlabel(axes1,'Fin Length (m)');
ylabel(axes1,'Fin Thickness (m)');
zlabel(axes1,'Number of Fins');
title(axes1, 'Solution Space for Heat Sink');
view(axes1,[-37.5 30]);
grid(axes1,'on');
hold(axes1,'all');
scatter3(valid_fin_length,valid_fin_thickness,valid_number_of_fins,50,'or');
scatter3(invalid_fin_length,invalid_fin_thickness,invalid_number_of_fins,50,'+
b');
legend1 = legend(axes1,'Valid Solution Space','Invalid Solution Space');

%% Case 1: Design & Analysis Valid results
[i,j]=find(Results(:,21)==1 & Results(:,22)==1);
dvalid_fin_length = Results(i,1);
dvalid_fin_thickness = Results(i,2);
dvalid_number_of_fins = Results(i,3);
dvalid_obj = Results(i,4);

%% Design Invalid - Analysis Valid results
[i,j]=find(Results(:,21)==0 & Results(:,22) == 1);
dinvalid_fin_length = Results(i,1);
dinvalid_fin_thickness = Results(i,2);
dinvalid_number_of_fins = Results(i,3);
dinvalid_obj = Results(i,4);

%% Design Invalid - Analysis Invalid results
[i,j]=find(Results(:,21) == 0 & Results(:,22)==0);
avalid_fin_length = Results(i,1);
avalid_fin_thickness = Results(i,2);
avalid_number_of_fins = Results(i,3);
avalid_obj = Results(i,4);

%% Design Valid - Analysis Invalid results
[i,j]=find(Results(:,21)==1 & Results(:,22)==0);
ainvalid_fin_length = Results(i,1);
ainvalid_fin_thickness = Results(i,2);
ainvalid_number_of_fins = Results(i,3);
ainvalid_obj = Results(i,4);

%% Create figure - plot design space for heat sink
figure3 = figure;
axes3 = axes('Parent',figure3);
xlabel(axes3,'Fin Length (m)');
ylabel(axes3,'Fin Thickness (m)');
zlabel(axes3,'Number of Fins');
view(axes3,[-37.5 30]);
grid(axes3,'on');
hold(axes3,'all');
scatter3(dvalid_fin_length,dvalid_fin_thickness,dvalid_number_of_fins,50,'or')
;
scatter3(dinvalid_fin_length,dinvalid_fin_thickness,dinvalid_number_of_fins,50
,'+b');
scatter3(avalid_fin_length,avalid_fin_thickness,avalid_number_of_fins,50,'*g')
;
scatter3(ainvalid_fin_length,ainvalid_fin_thickness,ainvalid_number_of_fins,50
,'xm');
legend3 = legend(axes3, 'Valid Design and Analysis ','Invalid Design, Valid
```

```
Analysis','Invalid Design and Analysis','Valid Design, Invalid Analysis');

%-------------------------------------------------------------------
%-------------------------------------------------------------------

function [R_total,h_bar] =
Heat_Sink_Resistance(fin_length,fin_width,Number_of_fins,
heat_sink_length,heat_sink_width,k_air,k_aluminum,heat_sink_base_thickness,cha
nnel_width)
%Author: Gregory Mocko
%Date: October 2005
% Calculation of fin resistance and fin array resistance for rectangular
extruded fins
%
%input
%   fin_length - length of the fin (meter)
%   fin_width - width of the fin (meter)
%   heat_sink_length - overall length of the heat sink (meter)
%   heat_sink_width - overall width of the heat sink (meter)
%   Number_of_fins - number of fins in the heat sink (#)
%   k_air - thermal conductivity of air (W/m-K)
%   k_aluminum - thermal conductivity of heat sink (W/m-K)
%   heat_sink_base_thickness - vertical thickness of heat sink base (meter)
%   channel_spacing - space between heat sink fins (meter)


%output
%   R_total - thermal resistance of heat sink (W/k)
%   h_bar - convection heat transfer coefficient (W/m^2)
%-------------------------------------------------------------------

L_c = fin_length + (fin_width/2);
A_fin = 2*heat_sink_length*L_c;

h_bar = Nusselt_interpolation(fin_length, channel_width,k_air);

A_p = L_c * fin_width;
A_b = channel_width * heat_sink_length;
A_t = Number_of_fins * A_fin + A_b;
mLc = ((2*h_bar)/(k_aluminum*A_p))^(1/2) * L_c^(3/2);
fin_efficiency = (tanh (mLc))/(mLc);

fin_array_efficiency = 1 - (Number_of_fins*A_fin/A_t)*(1 - fin_efficiency);
Resistance_total = 1 / (fin_array_efficiency*h_bar*A_t);
Resistance_conduction =
heat_sink_base_thickness/(k_aluminum*(heat_sink_length*heat_sink_width));
R_total = Resistance_total + Resistance_conduction;

%-------------------------------------------------------------------
%-------------------------------------------------------------------

function [P_critical] = Heat_Sink_Buckling(E_aluminum, fin_length,fin_width,
heat_sink_length)
%Author: Gregory Mocko
%Date: October 2005
%This function calculates the euler buckling load for the heat sink fin
%
%input
%   E_aluminum - modulus of elasticity of the aluminum heat sink (Pa)
%   fin_length - length of the fin (meter)
%   fin_width - width of the fin (meter)
%   heat_sink_length - overall length of the heat sink (meter)
```

```
%output
%   P_critical - critical load to cause buckling in heat sink fin (N)
%-------------------------------------------------------------------------
I_fin = 1/12*heat_sink_length*fin_width^3;
P_critical = pi^2*E_aluminum*I_fin / (4*fin_length^2);

%-----------------------------------------------------------------
%-----------------------------------------------------------------

function chip_temp =
Heat_Sink_Chip_Temp(chip_power,resistance,ambient_temperature);
%Author: Gregory Mocko
%Date: October 2005
%This function calculates the maximum temperature in the heat sink
%
%input
%   chip_power - total power dissipation of chip (Watts)
%   resistance - thermal resistance of heat sink (W/K)
%   ambient_temperature - temperature of surrounding environment (K)


%output
%   chip_temp - temperature at bottom of heat sink (K)
%-----------------------------------------------------------------

chip_temp = chip_power*resistance + ambient_temperature;

%-----------------------------------------------------------------
%-----------------------------------------------------------------

function [Stiffness_heat_sink] = Heat_Sink_Stiffness(E_aluminum,
fin_length,fin_width, heat_sink_length, heat_sink_width,Number_of_fins)
%Author: Gregory Mocko
%Date: October 2005
% Calculation of fin resistance and fin array resistance for rectangular
extruded fins
% The stiffness of the base is not considered in this calculation - it does
not affect the decision results because the base stiffness is constant
%input
%   E_aluminum - modulus of elasticity of the aluminum heat sink (Pa)
%   fin_length - length of the fin (meter)
%   fin_width - width of the fin (meter)
%   heat_sink_length - overall length of the heat sink (meter)
%   heat_sink_width - overall width of the heat sink (meter)
%   Number_of_fins - number of fins in the heat sink (#)

%output
%   Stiffness_heat_sink - vertical stiffness of heat sink fins (N/m)
%-------------------------------------------------------------------------
K_fin = (fin_width*heat_sink_length * E_aluminum)*Number_of_fins/fin_length;
Stiffness_heat_sink = K_fin;

%-----------------------------------------------------------------
%-----------------------------------------------------------------

function [fin_stress, fin_load] =
Heat_Sink_Stress(fin_width,heat_sink_length,Number_of_fins, Heat_Sink_Load);
%Author: Gregory Mocko
%Date: October 2005
%This function calculates the compressive stress in the fin
%
%   fin_width - width of the fin (meter)
```

```
%   heat_sink_length - overall length of the heat sink (meter)
%   Number_of_fins - number of fins in the heat sink (#)
%   Heat_Sink_Load - applied load to heat sink (N)

%output
%   fin_stress - compressive stress in heat sink fin (Pa)
%   fin_load - individual load on each fin
%-------------------------------------------------------------
fin_load = Heat_Sink_Load / Number_of_fins;
fin_stress = fin_load/(fin_width*heat_sink_length);

%-------------------------------------------------------------
%-------------------------------------------------------------

function [Heat_Sink_Space] = Heat_Sink_Volume(fin_length,heat_sink_length,
heat_sink_width, heat_sink_base_thickness)
%Author: Gregory Mocko
%Date: October 2005
%This function calculates the volume of the heat sink
%
%   fin_length - length of the fin (meter)
%   heat_sink_length - overall length of the heat sink (meter)
%   heat_sink_base_thickness - vertical thickness of heat sink base (meter)

%output
%   Heat_Sink_Space - volume of the heat sink (m^3)
%-------------------------------------------------------------
Area_heat_sink = heat_sink_width * (fin_length + heat_sink_base_thickness);
Heat_Sink_Space = Area_heat_sink * heat_sink_length;

%-------------------------------------------------------------
%-------------------------------------------------------------

function [Heat_sink_weight] = Heat_Sink_Weight(rho_aluminum,
fin_length,fin_width, heat_sink_length, heat_sink_width,
heat_sink_base_thickness,Number_of_fins)
%Author: Gregory Mocko
%Date: October 2005
% This function calculates the mass of the heat sink
% The mass of hte heat sink is actually calculated, not the weight

%   rho_aluminum - density of heat sink (kg/m^3)
%   fin_length - length of the fin (meter)
%   fin_width - width of the fin (meter)
%   heat_sink_length - overall length of the heat sink (meter)
%   heat_sink_width - overall width of the heat sink (meter)

%output
%   Heat_sink_weight - weight of the heat sink (kg)
%-------------------------------------------------------------
Heat_sink_weight =
rho_aluminum*((Number_of_fins*fin_length*fin_width*heat_sink_length)+(heat_sin
k_length*heat_sink_width*heat_sink_base_thickness));

%-------------------------------------------------------------
%-------------------------------------------------------------

function [h_bar] = Nusselt_interpolation(fin_length,channel_spacing,k_air)
%Author: Gregory Mocko
%Date: October 2005
%This function calculates the Nusselt Number and h_bar for the heat sink
% Reference: Incropera and Dewitt pg 450
% Assume completely developed laminar flow regime (Re < 2300)
```

```
%
%input
%   fin_length - length of the fin (meter)
%   channel_spacing - space between heat sink fins (meter)
%   k_air - thermal conductivity of air (W/m-K)


%output
%   h_bar - convection heat transfer coefficient (W/m^2)
%-----------------------------------------------------------------

aspect_ratio = fin_length/channel_spacing;

convection_coefficient (:,1) = [1.0; 1.43; 2.0; 3.0; 4.0;
8.0;15;20;25;30;35;40;45;50;55;60;65];
convection_coefficient (:,2) =  ;
Nu_value = spline(convection_coefficient(:,1), convection_coefficient(:,2),
aspect_ratio);

%channel_width = (heat_sink_width - Number_of_fins*fin_width) /
(Number_of_fins - 1);
A_channel = channel_spacing * fin_length;
P_channel = 2*(channel_spacing + fin_length);
Dh = 4*A_channel / P_channel;
h_bar = (k_air/Dh) * Nu_value ;
```

REFERENCES

[1] *OWL Web Ontology Language Overview* http://www.w3.org/TR/owl-features/

[2] *Phoenix Integration: Products: ModelCenter,* http://www.phoenix-int.com/products/ModelCenter.php

[3] 1991, *AIAA Technical Committee on Multidisciplinary Design Optimization (MDO): White Paper on Current State of the Art,* http://endo.sandia.gov/AIAA-MDOTC/sponsored/aiaa_paper.html

[4] 1999, *STEP: The Grand Experience: Special Publication 939,* National Institute of Standards and Technology, Gaithersburg, Maryland USA.

[5] 2005, *Process Specification Language (PSL),* http://www.mel.nist.gov/psl/

[6] 2005, *TechnoSoft Solutions,* www.technosoft.com

[7] October 2001, *ISO 10303-209: Composite and Metallic Structural Analysis and Related Design*, International Standards Organization (ISO).

[8] Aavid Thermalloy LLC, 2006, *Thermal Resistance Tool,* http://www.aavidthermalloy.com/products/bondfin/thermal.shtml

[9] Ahmed, S., S. Kim and K. M. Wallace, 2005, " A Methodology for Creating Ontologies for Engineering Design, Paper No. DETC2005-84729," *25th ASME Computers and Information in Engineering Conference (CIE)*, Long Beach, California USA.

[10] Alexandrov, N. M. and R. M. Lewis, 2004, "Reconfigurability in MDO Problem Synthesis, Part 1; Paper No. AIAA-2004-4307," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York USA.

[11] Alexandrov, N. M. and R. M. Lewis, 2004, "Reconfigurability in MDO Problem Synthesis, Part 2; Paper No. AIAA-2004-4308," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York USA.

[12] American Forest & Paper Association,2003, *Designing for Lateral-Torsional Stability in Wood Member, Technical Report 14*, American Forest & Paper Association, Inc.

[13] Arabshahi, S., D. C. Barton and N. K. Shaw, 1991, “Towards Integrated Design and Analysis,” *Finite Elements in Analysis and Design,* Vol. 9, No. 4, pp. 271-293.

[14] Arabshahi, S., D. C. Barton and N. K. Shaw, 1993, “Steps Towards CAD-FEA Integration,” *Engineering with Computers,* Vol. 9, No. 1, pp. 17-26.

[15] Armstrong, C. G., R. J. Donaghy and S. J. Bridgett, 1996, "Derivation of Appropriate Idealisations in Finite Element Modelling," *Advances in Finite Element Technology*, Civil-Comp Press, Edinburgh, United Kingdom, pp. 11 - 20.

[16] Baader, F., 2003, *The Description Logic Handbook : Theory, Implementation, and Applications* Cambridge University Press, Cambridge, UK.

[17] Bartholomew, P., 1998, "The Role of MDO within Aerospace Design and Progress Towards an MDO Capability, Paper No. AIAA-98-4705," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, Missouri USA.

[18] Bascaran, E., R. B. Bannerot and F. Mistree, 1989, "Hierarchical Selection Decision Support Problems in Conceptual Design," *Engineering Optimization,* Vol. 14, pp. 207-238.

[19] Belegundu, A. D. and T. R. Chandrupatla, 1999, *Optimization Concepts and Applications in Engineering,* Prentice Hall, Upper Saddle River, NJ.

[20] Bergmann, F., 2004, *Introduction to Description Logics,* http://www.fraber.de/sitec/dl.html

[21] Bohm, M., 2004, "Architecting Design Repositories to Support Design Knowledge Storage, Reuse and Design Tool Generation," *M.S. Thesis*, Department of Mechanical Engineering, University of Missouri - Rolla, Rolla, Missouri.

[22] Bohm, M. and R. Stone, 2004, "Product Design Support: Exploring a Design Repository System, IMECE2004-61746," *2004 ASME International Mechanical Engineering Congress*, Anaheim, California USA.

[23] Bohm, M. and R. Stone, 2004, "Representing Functionality to Support Reuse: Conceptual and Supporting Functions, Paper No. DETC2004/CIE-57693," *24th ASME Computers and Information in Engineering Conference (CIE)* Salt Lake City, Utah USA.

[24] Bohm, M., R. Stone and S. Szykman, 2003, "Enhancing Virtual Product Representations for Advanced Design Repository Systems, Paper No. DETC2003/CIE-48239," *23rd ASME Computers and Information in Engineering Conference (CIE)* Chicago, Illinois USA.

[25] Bollam, T., 1991, "Development of a Design Database for Decision Support Problems in a Concurrent Engineering Environment," *M.S. Thesis,* Department of Computer Science, University of Houston, Houston, TX.

[26] Bordiga, A., 1995, "Description Logics in Data Management," *Knowledge and Data Engineering,* Vol. 7, No. 5, pp. 671-682.

[27] Bordiga, A., 1996, "On the Relative Expressiveness of Description Logics and Predicate Logics," *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing Journal - Special Issue on Configuration,* Vol. 82, No. 1-2, pp. 353-367.

[28] Bras, B., Mistree, F., 1991, “Designing Design Processes in Decision-Based Concurrent Engineering,” *SAE Transactions Journal of Materials & Manufacturing*, pp. 451-458.

[29] Bras, B. A., 1992, "Foundations for Design Decision-Based Design Processes," *Ph.D. Dissertation*, University of Houston, Houston, Texas USA.

[30] Bronsvoort, W. F. and A. Noort, 2004, “Multiple-View Feature Modelling for Integrated Product Development,” *Computer Aided Design,* Vol. 36, No. 10, pp. 929-946.

[31] Brunnermeier, S. B. and S. A. Martin,1999, *99-1 Planning Report: Interoperability Cost Analysis of the U.S. Automotive Supply Chain*. Gaithersburg, Maryland USA, National Institute of Standards and Technology: 93.

[32] Burkett, W. C. and Y. Yang, 1995, “The STEP Integration Information Architecture,” *Engineering with Computers,* Vol. 11, pp. 136-144.

[33] Calvanese, D., M. Lenzerini and D. Nardi, 1998, "Description Logics for Data Modelling," *Logics for Databases and Information Systems*, Roma, Italy, pp. 229-263.

[34] Chen, B., D. Liu, B. Mahdavi, Q. Zhou, D. Bouhemhem, A. Ndiaye, F. Guibault, B. Ozell, D. Pelletier and J.-Y. Trepanier, 2001, "A Data-Centric Distributed Framework for MDO Management," *6th International Conference on Computer Supported Cooperative Work in Design*, London, Ontario, Canada. pp. 279-284.

[35] Chen, P., 1976, “The Entity-Relationship Model - Toward a Unified View of Data,” *ACM Transactions on Database Systems,* Vol. 1, No. 1, pp. 9-36.

[36] Chen, W., 1995, "A Robust Concept Exploration Method for Configuring Complex Engineering Systems," *Ph.D. Dissertation*, G.W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia USA.

[37] Chen, W., J. K. Allen, K. Tsui and F. Mistree, 1996, “A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors,” *ASME Journal of Mechanical Design,* Vol. 118, pp. 478-485.

[38] Chen, Y., 2001, "Computer-Aided Design for Rapid Tooling: Methods for Mold Design and Design-for-Manufacture," *Ph.D. Dissertation*, George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA.

[39] Choi, H.-J., R. Austin, J. Shepherd, J. K. Allen, D. L. McDowell, F. Mistree and D. L. Benson, 2004, "An Approach for Robust Micro-Scale Materials Design Under Unparameterizable Variability," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY.

[40] De Martino, T., B. Falcidieno and S. Habinger, 1998, “Design and Engineering Process Integration Through a Multiple View Intermediate Modeler in a Distributed Object-oriented System Environment,” *Computer Aided Design,* Vol. 30, No. 6, pp. 437-452.

[41] Eastman, C. M., A. H. Bond and C. S. C., 1991, “A Formal Approach for Product Model Information. ,” *Research in Engineering Design,* Vol. 2, No. 2, pp. 65-80.

[42] Efunda, 2005, *Columns: Inelastic Buckling,* http://www.efunda.com/formulae/solid_mechanics/columns/inelastic.cfm

[43] Fenves, S. J., 2001, *A Core Product Model for Representing Design Information, NISTIR 6736*. Gaithersburg, Maryland USA, National Institute of Standards and Technology (NIST): 38.

[44] Fenves, S. J.,2004, *CPM2: A Revised Core Product Model for Representing Design Information, NISTIR 7185*. Gaithersburg, Maryland USA, National Institute of Standards and Technology (NIST): 46.

[45] Fenves, S. J., R. D. Sriram, S. R. and F. Wang, 2005, "A Product Information Modeling Framework for Product Lifecycle Management," *Computer-Aided Design,* Vol. 37, No. 13, pp. 1399 - 1411.

[46] Fenves, S. J., C. Young, B. Gurumoorthy, G. M. Mocko and R. D. Sriram, 2003, *Master Product Model for the Support of Tighter Integration of Spatial and Functional Design, NISTIR 7004*. Gaithersburg, Maryland USA, National Institute of Standards and Technology (NIST): 28.

[47] Fernández, M. G., D. W. Rosen, J. K. Allen and F. Mistree, 2002, "A Decision Support Framework for Distributed Collaborative Design and Manufacture, Paper No. AIAA-4881," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, Georgia USA.

[48] Fernández, M. G., D. W. Rosen, J. K. Allen and F. Mistree, 2002, "Digital Interfaces: The Key to Effective Decision-Making in Distributed Collaborative Design and Manufacturing, Paper No. DETC2002/CIE-34466," *22nd ASME Computers and Information in Engineering Conference (CIE)*, Toronto, Canada.

[49] Fernández, M. G., C. C. Seepersad, D. W. Rosen, J. K. Allen and F. Mistree, 2001, "Utility-Based Decision, Support for Selection in Engineering Design, Paper No. DECT2001/DAC-21106," *27th ASME Design Automation Conference (DAC)*, Pittsburgh, PA. DETC2001/DAC-21106.

[50] Finn, D. P., 1993, "A Physical Modeling Assistant for the Preliminary Stages of Finite Element Analysis," *Artificial Intelligence in Engineering Design Manufacturing and Analysis,* Vol. 7, No. 4, pp. 231-237.

[51] Finn, D. P., J. B. Grimson and N. M. Harty, 1992, "An Intelligent Modelling Assistant for Preliminary Analysis in Design," *2nd International Conference on AI in Design*. pp. 579-596.

[52] Fragniere, E. and J. Gondzio, 2002, *Handbook of Applied Optimization,* Oxford University Press, New York, New York USA.

[53] Fulton, R. E. and B. Chadha,1999, *Engineering Data Management Course Notes, ME6754*. School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia USA.

[54] Gallaher, M. P. and A. C. O'Conner,2002, *Economic Impact Assessment of the International Standard for the Exchange of Product Model Data (STEP) in Transportation Equipment Industries*. Gaithersburg, Maryland USA, National Institute of Standards and Technology (NIST).

[55] Giesing, J. P. and J. M. Barthelemy, 1998, "A Summary of Industry MDO Applications and Needs, Paper No. AIAA 98-4737," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, Missouri USA.

[56] Gomez-Perez, A.,1999, *Ontological Engineering*. Tutorial on Ontological Engineering, IJCAI 99. Madrid, Spain, Laboratorio De Inteligencia Artificial, Universida Politecnica de Madrid.

[57] Gordon, S.,2001, *An Analyst's View: STEP-Enabled CAD-CAE Integration*. Pasadena, California USA, NASA's STEP for Aerospace Workshop.

[58] Grosse, I. R., *I-MAD: The Intelligent Modeling, Analysis, and Design Laboratory,* http://www.ecs.umass.edu/mie/labs/mda/fea/

[59] Grosse, I. R., J. M. Milton–Benoit and J. C. Wileden, 2005, "Ontologies for Supporting Engineering Analysis Models," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* Vol. 19, No. 1, pp. 1-18.

[60] Gruber, T. R., 1993, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal Human-Computer Studies,* Vol. 43, pp. 907-928.

[61] Gruber, T. R., J. M. Tenenbaum and J. C. Weber, 1992, "Toward a Knowledge Medium for Collaborative Product Development," *Artificial Intelligence in Design '92: Proceedings of the Second International Conference on Artificial Intelligence in Design*, Boston, Massachusetts.

[62] Hoffman, C. M. and R. J. Arinyo, 2000, "Distributed Maintenance of Multiple Product Views," *Computer Aided Design,* Vol. 32, No. 7, pp. 421-431.

[63] Hoffman, C. M. and R. Joan-Arinyo, 1998, "CAD and the Product Master Model," *Computer Aided Design,* Vol. 30, No. 11, pp. 905-918.

[64] Hunten, K. A., *AP209 EA Pilot Scenario*, Lockheed Martin Tactical Aircraft Systems.

[65] Hunten, K. A., 1997, *CAD/FEA Integration with STEP AP209 Technology and Implementation*, Lockheed Martin Corporation.

[66] Incropera, F. P. and D. P. DeWitt, 1996, *Fundamental of Heat and Mass Transfer,* John WIley & Sons, New York, New York.

[67] Isaacs, A., K. Sudhakar and P. M. Mujumdar, 2003, "Design and Development of MDO Framework, Paper No. 78," *International Conference on Modeling, Simulation, Optimization for Design of Multi-disciplinary Engineering Systems (MSO-DMES)*, Goa, India.

[68] Jha, K. and B. Gurumoorthy, 2000, "Automatic Propagation of Feature Modification Across Domains," *Computer Aided Design,* Vol. 32, pp. 691-706.

[69] Kamal, S. Z., H. M. Karandikar, F. Mistree and D. Muster, 1987, "Knowledge Representation for Discipline-Independent Decision Making," *Expert Systems in Computer-Aided Design* (J. Gero, Ed.), Elsevier Science Publishers B.V., Amsterdam, pp. 289-321.

[70] Karandikar, H. and F. Mistree, 1993, "Modeling Concurrency in the Design of Composite Structures," *Structural Optimization: Status and Promise* (M. P. Kamat, Ed.), AIAA, Washington, D.C., pp. 769-806.

[71] Karandikar, S., B. A. Bras, T. Bollam and F. Mistree, 1991, "Formulation, Storage, and Solution of Decision Support Problems in a Concurrent Engineering Environment," *CALS/CE Conference and Exposition*, Washington, DC.

[72] Kodiyalam, S. and J. Sobieszczanski-Sobieski, 2001, "Multidisciplinary design optimization - Some Formal Methods, Framework Requirements, and Application to Vehicle Design," *International Journal of Vehicle Design*, No. 1/2, pp. 3-22.

[73] Kopena, J. B., 2005, "Description Logic-Ground Knowledge Integration and Management," *Twentieth National Conference on Artificial Intelligence and the Seventeenth Annual Conference on Innovative Applications of Artificial Intelligence*, Pittsburgh, Pennsylvania USA. pp. 1612-1613.

[74] Kopena, J. B. and W. C. Regli, 2003, "Design Repositories On The Semantic Web With Description-Logic Enabled Services," *Proceedings of VLDB Semantic Web and Databases Workshop*, Humboldt-Universität, Berlin, Germany.

[75] Kopena, J. B. and W. C. Regli, 2003, "Functional Modeling of Engineering Designs for the Semantic Web," *IEEE Data Engineering Bulletin,* Vol. 26, No. 4, pp. 55-62.

[76] Kroo, I., 2004, *Distributed Multidisciplinary Design and Collaborative Optimization*. VKI Lecture Series on Optimization Methods and Tools for Multicriteria/Multidisciplinary Design Stanford, California USA, Stanford University.

[77] Kroo, I. and V. Manning, 2000, "Collaborative Optimization: Status and Directions, Paper No. AIAA-2000-4721," *8th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* Long Beach California, USA.

[78] Lee, J.-H. and H.-W. Suh, 2005, "OWL-Based Hybrid Product Knowledge Model for Collaborative Engineering Environment, Paper No. DETC2005/CIE-85574," *25th ASME Computers and Information in Engineering Conference (CIE)*, Long Beach, California USA.

[79] Levesque, H. J. and R. J. Brachman, 1987, "Expressiveness and Tractability in Knowledge Representation and Reasoning," *Computational Intelligence,* Vol. 3, pp. 7893.

[80] Lewis, K. and F. Mistree, 1997, "Collaborative, Sequential, and Isolated Decisions in Design, Paper No. DETC97/DTM-3883," *9th ASME International Conference on Design Theory and Methodology (DTM)*, Sacramento, California USA.

[81] Lewis, K. and F. Mistree, 1998, "The Other Side of Multidisciplinary Design Optimization: Accommodating a Multiobjective, Uncertain and Non-deterministic World," *Engineering Optimization,* Vol. 31, pp. 161-189.

[82] Liang, V.-C. and C. J. J. Paredis, 2004, "A Port Ontology for Conceptual Design of Systems," *Journal of Computing and Information Science in Engineering,* Vol. 4, No. 3, pp. 206-217.

[83] Lubell, J., R. S. Peak, V. Srinivasan and S. Waterbury, 2004, "STEP, XML, and UML: Complementary Technologies, Paper Number DETC 2004-57743" *24th ASME Computers and Information in Engineering Conference (CIE)*, Salt Lake City, Utah USA.

[84] Lyons, K., P. Hart, S. Shooter and W. Keirouz, 1998, "The Open Assembly Design Environment Project: An Architecture for Design Agent Interoperability, Paper No. DETC99/DFM-8945," *4th ASME Design for Manufacturing Conference (DFM)*, Las Vegas, Nevada.

[85] Malak, R. J. and C. J. J. Paredis, 2004, "On Characterizing and Assessing the Validity of Behavioral Models and their Predictions, Paper No. DETC2004/DTM-57452," *16th International Conference on Design Theory and Methodology (DTM)*, Salt Lake City, Utah USA.

[86] Mansingh, V. and E. Prather, *How to Design a Heat Sink,* http://www.icepak.com/prod/icepak/solutions/articles/iceart9.htm

[87] Mason, H., 2002, *ISO 10303 - STEP: A Key for the Global Market*. ISO Bulletin.

[88] McGuinness, D. L. and J. Wright, 1998, "Conceptual Modeling for Configuration: A Description Logic-based Approach," *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing Journal - Special Issue on Configuration,* Vol. 12, No. 4, pp. 333-344.

[89] Miller, J. G., 1978, *Living Systems,* McGraw-Hill, New York, New York.

[90] Mistree, F., B. A. Bras, W. F. Smith and J. K. Allen, 1996, "Modeling Design Processes: A Conceptual, Decision-Based Perspective," *International Journal of Engineering Design and Automation,* Vol. 1, No. 4, pp. 209-221.

[91] Mistree, F., O. F. Hughes and B. A. Bras, 1993, "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm," *Structural Optimization: Status and Promise* (M. P. Kamat, Ed.), AIAA, Washington, DC, pp. 247-286.

[92] Mistree, F., K. Lewis and L. Stonis, 1994, "Selection in the Conceptual Design of Aircraft," *5th AIAA/USAF/NASA/ISSMO Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*, Panama City, Florida USA. pp. 1153-1166

[93] Mistree, F., W. F. Smith and B. A. Bras, 1993, "A Decision-Based Approach to Concurrent Engineering," *Handbook of Concurrent Engineering* (H. R. Paresai and W. Sullivan, Eds.), Chapman & Hall, New York, pp. 127-158.

[94] Mistree, F., W. F. Smith, B. A. Bras, J. K. Allen and D. Muster, 1990, "Decision-Based Design: A Contemporary Paradigm in Ship Design," *Transactions, Society of Naval Architects and Marine Engineers,* Vol. 98, pp. 565-597.

[95] Mistree, F., W. F. Smith, S. Z. Kamal and B. A. Bras, 1991, "Designing Decisions: Axioms, Models and Marine Applications," *Fourth International Marine Systems Design Conference*, Kobe, Japan. pp 1-24.

[96] Mocko, G. and S. Fenves,2003, *A Survey of Design-Analysis Integration Issues, NISTIR 6996*. Gaithersburg, Maryland USA, National Institute of Standards and Technology (NIST): 47.

[97] Mocko, G., R. Malak, C. Paredis and R. S. Peak, 2004, "A Knowledge Repository For Behavioral Models in Engineering Design, Paper No. DETC/CIE 2004-57746," *24th ASME Computers and Information in Engineering Conference (CIE)*, Salt Lake City, Utah USA.

[98] Murdock, J. W., S. Szykman and R. D. Sriram, 1997, "An Information Modeling Framework to Support Design Databases and Repositories, Paper No. DETC1997/DFM-4373," *2nd ASME Design for Manufacturing Conference (DFM)*, Sacramento, California.

[99] Muster, D. and F. Mistree, 1988, "The Decision Support Problem Technique in Engineering Design," *International Journal of Applied Engineering Education,* Vol. 4, No. 1, pp. 23-33.

[100] Mylopoulos, J., 1998, "Information Modeling in the Time of the Revolution," *Information Systems,* Vol. 23, No. 3-4.

[101] Nanda, J., T. Simpson, S. B. Shooter and R. Stone, 2005, "A Unified Information Model for Product Family Design Management, Paper No. DETC2005-84869," *25th ASME Computers and Information in Engineering Conference (CIE)*, Long Beach, California USA.

[102] Navathe, S. and R. Elmasri, 2004, *Fundamentals of Database Systems* Pearson/Addison Wesley, Boston.

[103] Ndiaye, A., F. Guibault, B. Ozell and J.-Y. Trepanier, 2000, *The VADOR Project: A Step Towards an MDO Capability,* http://www.cfdsc.ca/english/bulletins/11/1110.html

[104] Noy, N. and D. L. McGuinness, 2001, *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford, California USA, Knowledge Systems Laboratory, Stanford University.

[105] Pahl, G. and W. Beitz, 1999, *Engineering Design: A Systematic Approach,* Springer, New York, New York.

[106] Pan, J. Z., 2004, "Description Logics: Reasoning Support for the Semantic Web," *Ph.D. Dissertation*, School of Computer Science, University of Manchester, Manchester.

[107] Panchal, J., 2005, " A Framework for Simulation-Based Integrated Design of Multiscale Products and Design Processes," *Ph.D. Dissertation,* George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA.

[108] Panchal, J. H., M. G. Fernández, C. J. J. Paredis, J. K. Allen and F. Mistree, 2004, "Designing Design Processes in Product Lifecycle Management: Research Issues and Strategies, Paper No. DETC2004/CIE-57742," *24th ASME Computers and Information in Engineering Conference (CIE)*, Salt Lake City, Utah USA.

[109] Panchal, J. H., M. G. Fernández, C. J. J. Paredis and F. Mistree, 2004, "Reusable Design Processes via Modular, Executable, Decision-Centric Templates, Paper No. AIAA-2004-4601," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York USA.

[110] Paredis, C. J. J., A. Diaz-Calderon, R. Sinha and P. K. Khosla, 2001, "Composable Models for Simulation-Based Design," *Engineering with Computers,* Vol. 17, No. 2, pp. 112-128.

[111] Patil, L., D. Dutta and R. D. Sriram, 2005, "Ontology Formalization of Product Semantics for Product Lifecycle Management, Paper No. DETC2005-85121," *25th ASME Computers and Information in Engineering Conference (CIE)*, Long Beach, California.

[112] Peak, R. S., 1993, "Product Model-Based Analytical Models (PBAMS): A New Representation of Engineering Analysis Models," *Ph.D. Dissertation,* G.W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia USA.

[113] Peak, R. S., 2003, "Characterizing Fine Grained Associativity Gaps: A Preliminary Study of CAD-CAE Model Interoperability, Paper No. DETC2003/CIE-48232," *23rd ASME Computers and Information in Engineering Conference (CIE)* Chicago, Illinois.

[114] Peak, R. S., R. E. Fulton, I. Nishigaki and N. Okamoto, 1998, "Integrating Engineering Design and Analysis Using a Multi-Representation Approach," *Engineering with Computers,* Vol. 14, No. 2, pp. 93-114.

[115] Peak, R. S., J. Lubell, V. Srinivasan and S. Waterbury, 2004, "STEP, XML, and UML: Complementary Technologies," *Journal of Computing and Information Science in Engineering (Transactions of the ASME)*, Vol. 4, No. 4, pp. 379-390.

[116] Peak, R. S., A. J. Scholand, D. R. Tamburini and R. E. Fulton, 1999, "Towards the Ubiquitization of Engineering Analysis to Support Product Design," *International Journal of Computer Applications in Technology,* Vol. 12, No. 1, pp. 1-15.

[117] Pederson, K., J. Emblemsvag, J. K. Allen and F. Mistree, 2000, "Validating Design Methods and Research - The Validation Square, Paper No. DETC00/DTM-14579," *12th ASME International Conference on Design Theory and Methodology (DTM),* Baltimore, Maryland USA.

[118] Prabhakar, V. and S. D. Sheppard, 1994, "A Knowledge-Based Approach to Model Idealization in FEM," *10th Conference on Artificial Intelligence for Applications*, San Antonio, Texas USA.

[119] Radian Heatsinks, 2004, *25mm Plate Fin BGA Heat Sink - HS1802EB,* http://www.radianheatsinks.com/standard/hs1802eb.html

[120] Rajan, V., *Position Paper: Decision-Based Design,* http://www.eng.buffalo.edu/Research/DBD/papers/rajan.html

[121] Regli, W. C., *National Design Repository,* http://edge.cs.drexel.edu/repository/frameset.html

[122] Regli, W. C., C. Foster, E. Hayes, C. Y. Ip, D. McWherter, M. Peabody, Y. Shapirsteyn and V. Zaychik, *National Design Repository Project: A Status Report*. Philadelphia, Pennsylvania USA, Drexel University.

[123] Reynolds, D., 2001, *Semantic Web Chalk Talk: Amateur Intro to Description Logics*.

[124] Robinson, S., R. E. Nance, R. J. Paul, M. Pidd and S. J. E. Taylor, 2004, "Simulation Model Reuse: Definitions, Benefits, and Obstacles," *Simulation Modeling Practice and Theory,* Vol. 12, pp. 479-494.

[125] Rosen, D., Y. Chen, J. Gerhard, J. K. Allen and F. Mistree, 2000, "Design Decision Templates and Their Implementation for Distributed Design and Solid Freeform Fabrication, Paper No. DETC00/DAC-14293," *25th ASME Design Automation Conference (DAC)*, Baltimore, Maryland USA.

[126] Rosen, D., Y. Chen, S. Sambu, J. K. Allen and F. Mistree, 2003, "The Rapid Tooling Testbed: A Distributed Design-for-Manufacturing System," *Rapid Prototyping Journal,* Vol. 9, No. 3, pp. 122-132.

[127] Rosenman, M. A. and J. S. Gero, 1996, "Modelling Multiple Views of Design Objects in a Collaborative CAD Environment," *Computer-Aided Design*, Vol. 28, No. 3, pp. 193-205.

[128] Rosenman, M. A. and J. S. Gero, 1997, "Collaborative CAD Modelling in Multidisciplinary Design Domains," *Formal Aspects of Collaborative Computer-Aided Design*, Key Centre of Design Computing, University of Sydney, Sydney Australia. pp. 387-404.

[129] Salas, A. O. and J. C. Townsend, 1998, "Framework Requirements for MDO Application Development, Paper No. AIAA-98-4740," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, Missouri USA.

[130] Sambu, S., T. Chen and D. Rosen, 2003, “Geometric Tailoring: A Design For Manufacturing Method for Rapid Prototyping and Rapid Tooling,” *ASME Journal of Mechanical Design,* Vol. 126, No. 4, pp. 571-580.

[131] Sambu, S. P., 2001, "A Design for Manufacture Method for Rapid Prototyping and Rapid Tooling," *M.S. Thesis,* GWW School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia.

[132] Schnell, A., 2004, "Using the Compromise Decision Support Problem in Microsystem Design: A Formulation for a Miniature Parylene Gas Chromatographic Column, Paper No. DETC2004/DTM-57582," *16th ASME Design Theory and Methodology Conference (DTM)*, Salt Lake City, Utah.

[133] Seepersad, C. C., 2001, "The Utility-Based Compromise Decision Support Problem with Applications in Product Platform Design," *M.S. Thesis*, The G.W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA.

[134] Sheehy, D. J., C. G. Armstrong and R. D.J., 1995, "Computing the Medial Surface of a Solid from A Domain Delauney Triangulation," *ACM/IEEE Symposium on Solid Modeling and Applications*, Salt Lake City, Utah USA. pp. 201-212.

[135] Sheffer, A., 2001, “Model Simplification for Meshing Using Face Clustering,” *Computer Aided Design,* Vol. 33, pp. 925-934.

[136] Shephard, M. S., P. L. Baehmann, M. K. Georges and E. V. Korngold, 1990, “Framework for the Reliable Generation and Control of Analysis Idealizations,” *Computer Methods in Applied Mechanics and Engineering,* Vol. 82, pp. 257-280.

[137] Shephard, M. S. and R. Wentorf, 1994, “Toward the Implementation of Automated Analysis Idealization Control,” *Applied Numerical Mathematics,* Vol. 14, pp. 105-124.

[138] Shooter, S. B., W. T. Keirouz, S. Szykman and S. J. Fenves, 2000, "A Model for Information Flow in Design, Paper No. DETC2000/DTM-14550," *12th ASME International Conference on Design Theory and Methodology (DTM)*, Baltimore, Maryland USA.

[139] Sim, S. K. and A. H. B. Duffy, 2003, “Towards an ontology of Generic Engineering Design Activities,” *Research in Engineering Design,* Vol. 14, No. 4, pp. 200-223.

[140] Simon, H. A., 1996, *The Sciences of the Artificial,* MIT Press, Cambridge, Massachusetts.

[141] Sobieszczanski-Sobieski, J. and R. T. Haftka, 1997, “Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments,” *Structural and Multidisciplinary Optimization,* Vol. 14, pp. 1-23.

[142] Soltis, L. A., 1999, "Structural Analysis Equations," *Wood Handbook: Wood as an Engineering Material*, USDA Forest Service, Forest Products Laboratory, Madison, Wisconson USA, pp. 8.1-8.11.

[143] Stone, R., *Design Engineering Lab Repository,* http://function2.basiceng.umr.edu:8080/view/options.jsp

[144] Svensson, D., J. Malmstrom, P. Pikosz and J. Malmqvist, 1999, "A Framework for Modelling and Analysis of Engineering Information Management Systems, DETC99/CIE-9006," *19th ASME Computers and Information in Engineering Conference (CIE)*, Las Vegas, Nevada USA.

[145] Szykman, S., 2002, "Architecture and Implementation of a Design Repository System, Paper No. DETC2002/CIE-34463," *22nd ASME Computers and Information in Engineering Conference (CIE)*, Montreal, Canada.

[146] Szykman, S., S. B. Shooter, S. F. Fenves and W. Keirouz, 2001, "A Foundation for Interoperability in Next-Generation Product Development Systems," *Computer-Aided Design,* Vol. 33, pp. 545-559.

[147] Szykman, S., R. D. Sriram, C. Bochenek and J. Racz, 1999, "The NIST Design Repository Project," *Advances in Soft Computing - Engineering Design and Manufacturing* (R. Roy , T. Furuhashi and P. K. Chawdhry, Eds.), Spring-Verlag, London, pp. 5-19.

[148] Szykman, S., R. D. Sriram, C. Bochenek, J. W. Racz and J. Senfaute, 2000, "Design Repositories: Engineering Design's New Knowledge Base," *IEEE Intelligent Systems,* Vol. 15, No. 3, pp. 48-55.

[149] Szykman, S., R. D. Sriram, C. Bochenek and J. Senfaute, 2000, "Design Repositories: Next-Generation Engineering Design Databases," *IEEE Intelligent Systems and Their Applications*, Vol. 15, No. 3, pp. 48-55.

[150] Tautges, T. J., 2001, "The Generation of Hexahedral Meshes for Assembly Geometry: Survey and Progress," *International Journal for Numerical Methods in Engineering,* Vol. 50, No. 12, pp. 2617-2642.

[151] Tautges, T. J., T. Blacker and S. A. Mitchell, 1996, "The Whisker Weaving Algorithm: A Connectivity-Based Method for Constructing All-Hexahedral Finite Element Meshes," *International Journal for Numerical Methods in Engineering,* Vol. 39, pp. 3327-3349.

[152] Taylor, B. N. and P. J. Mohr, 1998, *The NIST Reference on Constants, Units, and Uncertainty,* http://physics.nist.gov/cuu/Units/

[153] TechnoSoft Inc, 2003, *The Adaptive Modeling Language. A Technical Perspective,* www.technosoft.com/docs/AML-Technical-Perspective.pdf

[154] Townsend, J. C., J. A. Samareh, R. P. Weston and W. E. Zorumski, 1998, *Integration of a CAD System into an MDO Framework, NASA/TM-1998-207672*. Hampton, Virginia USA, NASA-Langley Research Center: 13.

[155] Tummala, R., 2001, *Fundamentals of Microsystems Packaging,* McGraw-Hill, New York.

[156] Uschold, M., King, M. 1995. *Towards a Methodology for Building Ontologies AIAI-TR-183*. Edinburgh, UK, Artificial Intelligence Applications Institute.

[157] Vander Kam, J. C., P. Gage and J. Sohn, 2004, "The Launch Vehicle Language Data Model Applied to Analysis Tool Connectivity, Paper No. AIAA-2004-0206," *42nd AISAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada USA.

[158] Weiss, K. A., E. C. Ong and N. G. Leveson, 2003, "Reusable Specification Components for Model-Driven Development," *International Conference on System Engineering (INCOSE '03)*, Crystal City, Virginia USA.

[159] Weiss, K. A., E. C. Ong and N. G. Leveson, 2003, "Reusable Software Architectures for Aerospace Systems," *Aircraft Engineering and Aerospace Technology,* Vol. 75, No. 5, pp. 461-469.

[160] Weisstein, E. W., 2005, *"Exhaustive Search." From MathWorld--A Wolfram Web Resource.,* http://mathworld.wolfram.com/ExhaustiveSearch.html

[161] Wentorf, R. and M. S. Shephard, 1993, "User Interface Functions for an Analysis Idealization System," *Microcomputers in Civil Engineering,* Vol. 8, pp. 85-95.

[162] Whitney, D. E., 1995, *State of the Art in the United States of CAD Methodologies for Product Development*. Cambridge, Massachusetts USA, Center for Technology, Policy, and Industrial Development, Massachusettes Institute of Technology: 35.

[163] Wierzbicki, A. P., M. Marek and J. Wessels, 2000, *Model-Based Decision Support Methodology with Environmental Applications* Kluwer Academic Publishers, Boston.

[164] Willcox, K., 2004, *Multidisciplinary System Design Optimization (MSDO): Multidisciplinary Design and Analysis Problem Formulation: Lecture 2*: 35.

[165] Wilson, M. W., 2000, "The Constrained Object Representation for Engineering Analysis Integration," *M.S. Thesis,* G.W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA.

[166] Xiao, A., 2003, "Collaborative Multidisciplinary Decision Making in a Distributed Environment," *Ph.D. Dissertation,* G.W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia USA.

[167] Xiao, A., H.-J. Choi, J. K. Allen, D. Rosen and F. Mistree, 2001, "A Web-Based Distributed Product Realization Environment, Paper No. DETC2001/CIE-21766," *21st ASME Computers and Information in Engineering Conference (CIE)*, Pittsburgh, Pennsylvania USA.

[168] Xiao, A., S. Zeng, J. Allen, F. Mistree and D. W. Rosen, 2002, "Collaborating Multidisciplinary Decision Making Using Game Theory and Design Capability Indices, Paper No. AIAA 2002-5617," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, Georgia USA.

[169] Yoshioka, M. and T. Tomiyama, 1997, "Pluggable Metamodel Mechanism: A framework of an Integrated Design Object Modeling Environment," *1997 Lancaster International Workshop on Engineering Design (CACD)*, Lancaster, UK. pp. 57-70.

[170] Zolin, E., *Complexity of reasoning in description logics,* http://www.cs.man.ac.uk/~ezolin/logic/complexity.html

[171] Zweber, J. V., M. Blair, H. Kamhawi, G. Bharatram and A. Hartong, 1998, "Structural and Manufacturing Analysis of a Wing Using the Adaptive Modeling Language, Paper No. AIAA-1998-4869," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, Missouri USA.

VITA

Gregory M. Mocko was born in Rockville, Connecticut in 1976. He grew up in Stafford Springs Connecticut and completed his high school education in the same town in 1994. He attended the University of Connecticut from 1994-1999 and earned Bachelor of Science degrees in Mechanical Engineering and Material Science. After his undergraduate education he joined the graduate program in Mechanical Engineering at Oregon State University. He received his Master of Science degree in Mechanical Engineering in the field of Engineering Design in 2001. After finishing at Oregon State University he joined the G.W. Woodruff School of Mechanical Engineering at the Georgia Institute of Technology in 2001. He received his PhD in Mechanical Engineering in 2006 in the field of Engineering Design and Computer-Aided Engineering. After his PhD, Greg will join the faculty as an assistant professor in the Department of Mechanical Engineering at Clemson University. His research interests include computer supported design environments for distributed and multi-disciplinary decision making and engineering information technology.